Face value of companies: deep learning for nonverbal communication

Master's Thesis submitted

to

Prof. Dr. Wolfgang Karl Härdle

Prof. Dr. Cathy Yi-Hsuan Chen

Humboldt-Universität zu Berlin

School of Business and Economics

Institute for Statistics and Econometrics

Ladislaus von Bortkiewicz Chair of Statistics

by

Sophie Burgard

564569

in partial fulfillment of the requirements

for the degree of

Master of Science

Berlin, November 24, 2017

Abstract

As a side effect of digitalization, a massive amount of unstructured data is generated every day. Unstructured data comprises video, speech, text, and image data, which are easy to interpret for humans - but can be challenging for computers. Financial research has been much engaged in recent history with decision-making based on textual or sentiment analysis. Textual analysis is based on the verbal part of communication, but in human interaction on a face-to-face level, nonverbal communication can play an equally important role in supporting a message. The interpretation of emotions in facial expressions is a major component of nonverbal communication. Deep learning is a versatile technique, that is used in numerous applications, providing somewhat cognitive capabilities for machines. This thesis describes how to build a deep convolutional neural network with the ability to detect emotions in faces. Different approaches in deep convolutional model designs are tested and evaluated. The results are then used to evaluate videos of the regular press conference of the European Central Bank between January 2011 and September 2017. This processing step results in emotional-scores of facial expressions from 70 press conferences and more than 200,000 single pictures. It is investigated whether information of nonverbal communication, measured in levels of emotional excitement, can be linked to the movements of the Euro Stoxx 50 index. This 'face value' is compared to the value of speech and accompanying research. Using image data from press conferences as source of unstructured data and transferring of nonverbal communication to stock markets are both topics that, to the best of found knowledge, have not yet been focused upon in research.

Keywords: convolutional neural network, deep learning, returns, financial, empirical, unstructured data, multivariate analysis

Contents

List of Figures

List of Abbreviations

ANN artificial neural network

API application programming interface

Adam adaptive moments (estimation)

CNN convolutional neural network

CE cross-entropy

CET Central European Time

DL deep learning

DA discriminant analysis

ECB European Central Bank

FC fully connected (layer)

FER facial expression recognition

GD gradient descent

KL Kullback-Leibler (divergence)

MLP multilayer perceptron

NN neural network

OLS ordinary least squares

PC press conference

PCA principal component analysis

PLS partial least squares

ReLU rectified linear unit

RBG red green blue

SE squared error

SGD stochastic gradient descent

VGG visual geometry group

1 Introduction

As a side effect of digitalization, a massive amount of unstructured data is generated every day. Unstructured data comprises images, speech, text, and increasingly video data. For a human being, the interpretation of such data sources is usually a simple task - for a machine it can be challenging. But computers with somewhat human-like cognitive capabilities are not anymore visions of a remote future. This development is strongly linked to deep learning networks. Tesla's self-driving cars are able to observe their surroundings and make their decisions based on deep learning (Helbing, 2015), IBM's Watson has almost human-like capabilities in processing speech (Feng et al., 2015), only naming some examples of the versatile and various applications this technology can master.

Apart from the field of robotics deep learning techniques have made their way into many scientific domains. The research of textual data, or sentiment analysis, has been given much attention in recent years. In the area of financial research, text mining is a large field of interest, due to the various possibilities of incorporating information from unstructured data into decision making (Kumar and Ravi, 2016). Textual analysis is based on the verbal part of communication, but in a face-to-face setting the understanding and interpretation of a message can also be highly dependent on a nonverbal factor. Nonverbal communication is a combination of body posture, facial expressions (happy, sad, surprised, etc.), gestures and also an audible component. The remainder of this thesis will use the recognition of facial expressions as measure of nonverbal communication, because it is straightforward in categorization and detection.

Ambition of this thesis is to link nonverbal communication, as it can be observed during live broadcasts of press conferences, to financial measures. The question of research is, if it is possible to generate a similar signal from observing facial expressions to the stock market, as it can be seen in sentiment analysis. This effect will be considered as the 'face value of a company'. Why it might be beneficial to observe nonverbal communication for the analysis of live streams is based on the following hypotheses: A facial expression, or subtle movements, are harder to fake and more spontaneous than words, they therefore might contain a more honest level of information (Porter and ten Brinke, 2008). The interpretation of an emotion displayed on a person's face can be made without knowledge of the person, the language of the speaker, and also without knowledge of the topic. To some extent nonverbal communication can be considered as universal (Ekman and Friesen, 1971). A facial expression might be therefore a more versatile measure than speech.

This paper splits the empirical analysis into two major parts. First, it is necessary to chose a deep convolutional neural network design, that has the capacity to interpret faces. This is done by comparing different designs for the task of facial expression recognition. Then the different options are applied and evaluated using a suitable data set. In computer vision the automated interpretation of faces has become an interesting field of research (Kumari et al., 2015). Applications for facial expression recognition are used in marketing for evaluating consumer reactions (Shergill et al., 2008) or measuring the joy of players during computer games (Zhan et al., 2008).

The second part of the empirical analysis is engaged with connecting facial expressions during a live press conference to financial measures. This paper will use the regular press conferences of the European Central Bank's president, because they offer a good experimental setting for this analysis. Extensive research has focused on the textual analysis in this area, for example: Rosa (2011) investigate the importance of the sentiment of central bank communications on global stock market volatilities, Moniz and de Jong (2014) use the speech sentiment to predict the interest rate expectations. The investigations have been also engaged on intraday levels (Hussain, 2011) and also to measure long-term effects of speech sentiment on the stock markets (Schmeling and Wagner, 2015).

To the best found knowledge, this thesis is new in three aspects: Video data and press conferences have not yet been used as a source of unstructured data in financial research, exceeding the textual analysis of such events. Treating facial expression recognition as a measure of nonverbal communication using deep learning methods is a new approach. Applying nonverbal communication in a context of finance, that is not on a consumer confidence level in a marketing context, has not yet been focused upon in research.

The remainder of this thesis is as follows: Section two provides a substantial overlook over the different elements, that are needed to build a deep convolutional neural network, describing different layer types, modern optimization procedures and architectural possibilities. A special focus will be put on the task of image classification. Section three uses a Kaggle dataset on facial expression recognition and transfers the theoretical model considerations into practice. Three different approaches on emotion-classification are applied and evaluated. Section four is transferring the potential of facial expression recognition, as a component of nonverbal communication, and measures its effect on the stock market during live events of companies and

institutions. After discussing requirements on the data and the considered events, 70 single press conferences of the European Central Bank between 2011 and 2017 are reviewed. Different estimation methods are used to detect a potential link between nonverbal communication, expressed as facial expressions, and the stock market.

Additional material to this thesis can accessed under www.quantlet.com and is indicated by the Quantlet-symbol .

2 Deep learning

2.1 Artificial neural networks

Deep feedforward network neural networks, also sometimes called multilayer perceptrons (MLP), are the most important class of deep learning models. Neural networks (NN) are a mathematical attempt to recreate the information processing process of the brain. The general purpose of a NN is mapping some input $\mathbf{x}$ to some prespecified output $\mathbf{y}$. 'Feedforward' describes the one-directional (acyclic) flow of information through the network. Unlike recurrent (cyclic) NNs, feedforward networks do not contain feedback loops. The net-like structure of NNs arises from the chaining of many nonlinear functions in so-called layers. The input of one layer is the output of the preceding layer. The depth of a network describes the number of stacked layers in a network . For some input $\mathbf{x}$ and some output $\mathbf{y}$ a NN with depth three could be written as

$$\mathbf{y} = f^4(f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x}))), \tag{1}$$

where $f^{(1)}$ is the first or input layer, $f^{(4)}$ is the fourth or in this case output layer. The intermediate functions $f^{(2)}$ and $f^{(3)}$ are hidden layers. The special properties and almost universal capabilities of NNs arise from adaptive weights throughout the network structure. A deep network is commonly any network with more than one hidden layer. An intuitive way of visualizing NNs is displaying them in form of flow graphs, see Fig. 1. A network as in Eq. (1) is called three layer NN. By convention only layers with adjustable weights are counted. The weights of the input layer are per definition fixed.

The further description of the theory will be limited to modeling classification problems, since this is in line with the research question of this paper. Nevertheless also regression problems can be solved by using deep neural networks.

2.1.1 Single neuron

A neural network consists of different layers, each layer consists of many single units, the neurons. One neuron is the smallest processing unit of the network (circular structures in Fig. 1). It receives an input, either raw data or some output from another unit, transforms it by some nonlinear function and passes it on. These nonlinear functions are called activation functions. The final output results into the scores which allow to make a classification decision.

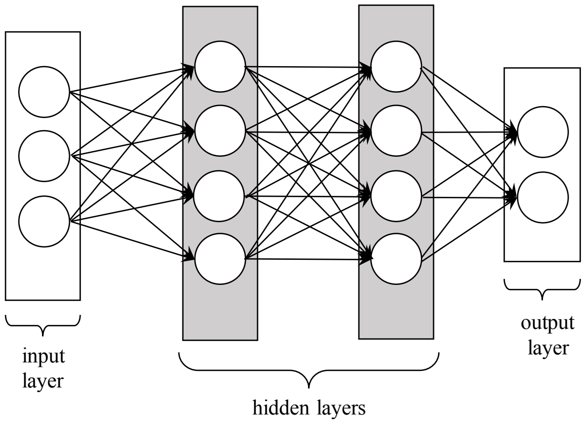


Figure 1: Fully connected three layer neural network

A single neuron performs the following steps during the feed forward process (see Fig. 2): First it collects all input values x_i, multiplies them with their respective weights w_i and adds a bias term e. Second, a nonlinear activation function is applied to the weighted sum of the input factors. This result will be passed on by each neuron to the subsequent layers. Each element such as x_i, y, e or y_i can be a scalar, a vector or a matrix, depending on the network's design. Adjusting the weights w_i and biases e for each neuron in every layer is the task of network training.

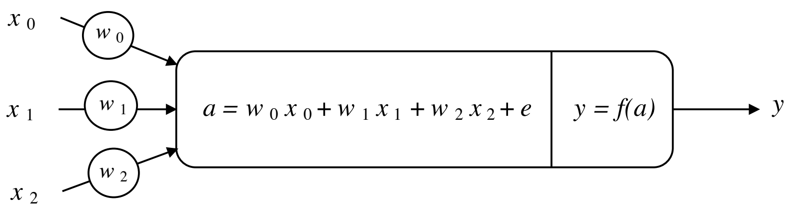


Figure 2: Information processing of a single neuron, $f(\cdot)$ some activation function

2.1.2 Activation functions

Nonlinear activation functions are crucial element of the design of a NN. The nonlinearity is of importance, since a network that only consists out of linear functions, is a linear transformation itself. Its modeling power is therefore restricted to linear dependencies, which is an impractical limitation for most problems. It is not necessary to define the specific form of the nonlinear dependency in a neural network that perfectly fits the problem. Neural networks with at least one hidden layer and a nonlinear activation function are a universal approximation system. The "universal approximation theorem (Cybenko, 1989; Hornik, 1991) states that a feedforward network with a linear output and at least one hidden layer with any 'squashing' (for example logistic sigmoid activation function) can approximate any Borel measurable function from one finite-dimensional space to another with any desired non-zero amount of error, provided that the network is given enough hidden units." (Goodfellow et al. (2016), p. 198). The same can be proven for rectified linear units (Leshno et al. (1993)) and other common forms of nonlinear activation functions such as maxout (Goodfellow et al. (2013)). Since activation functions are essential parts of NNs, there is constant research about new possibilities, which are both effective in shaping functional forms and also allow efficient training.

Most common for neural networks are rectified linear units for hidden units. For output layers, depending on the task, sigmoid or softmax-activation functions are used, which both derive from the same statistical background. The following paragraphs discuss the use of different variations of common activation functions.

Sigmoid Function The sigmoid function or logistic function

$$\sigma(x) = \frac{1}{1 + exp(-x)}$$

shows a 's'-shaped form (Fig. 3a), with support of all real-valued numbers and an output limited to the interval of $(0, 1)$. It used to be in favor in early applications of neural networks due to the analogy and interpretation as the firing rate of a biological neuron: at zero no signal is passed on, at one the neuron fires at full rate.

The state-of-art procedure for optimization processes of NNs is using some gradient descent method in combination with back-propagation for gradient calculation. The sigmoid function has fallen out of favor due to the following undesirable properties when using gradient based optimizers (Hochreiter et al., 2009):

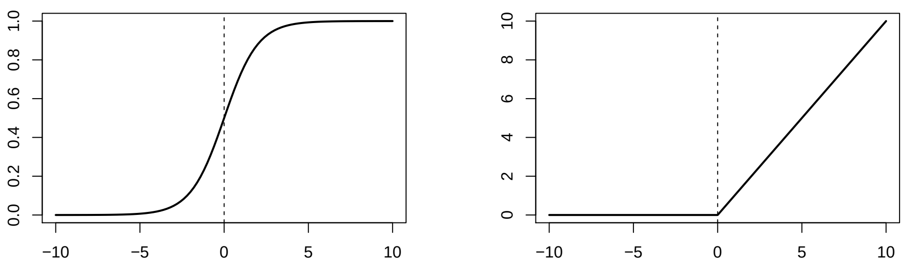
- vanishing gradient problem:
  The gradient in the tails of the sigmoid function converges quickly to zero. The optimization procedure updates the weights in direction of the gradient. If the gradient is almost saturated at the tails, the optimization of the weights is inefficient or can be even impossible within finite training time.

- non-zero centering of output:
  This property induces undesired dynamics into the weight updating process, because all gradients are constantly negative or positive.

These problems can be overcome by a careful choice of the initial weights when starting the optimization process or by choosing a different form of activation functions for the hidden layers.

The sigmoid function is still a necessary component, if the NN's target is training a binary classifier. It is used as the activation function of the output layer. Statistically speaking, this is the same as applying the a logisitic regression to the input coming from the preceding layers. The output of the sigmoid function can be directly interpreted as the probability of belonging to one of two possible groups.

A different form based on the sigmoid function is the hyperbolic tangent $\tanh(x) = 2\sigma(2x) - 1$, where $\sigma(\cdot)$ stands for the sigmoid function. It maps all real-valued input to the range (-1,1). Its output is zero-centered, which is an advantage compared to the sigmoid function, but still the vanishing gradient problem can occur, which is the more serious problem.



(a) Sigmoid (b) Rectified linear unit (ReLU)

Figure 3: Common forms of nonlinear activation functions for hidden layers

Softmax activation In a statistical context, a softmax activation is the same as estimating a a multiclass logit regression to the output of the preceding layer. It is the multiclass analogy to the interpretation of the sigmoid function as a binary classifier. This interpretation can be derived from the maximum likelihood principle, assuming that each representation of the output is coming from an exponential distribution, for more detail see e.g. Duda et al. (2001, p. 27). The probability of belonging into a specific class j, which is one of overall K classes, is given by

$$\sigma_j(\mathbf{x}) = \frac{\exp(x_j)}{\sum_{k=1}^{K} \exp(x_k)}$$

It holds that $\sum_{i=1}^{K} \sigma_i = 1$ and same time $0 \leq \sigma_i \leq 1$ for any i.

Rectified linear unit Rectified linear units, commonly shortened as ReLU,

$$f(x) = \max(0, x)$$

have become the default recommendation for activation functions in modern neural networks (e.g. Jarrett et al., 2009). ReLUs can greatly speed up the computational time of the optimization process, when using stochastic gradient optimization methods. Due to the strict cut-off at zero (Fig. 3b), ReLus are exposed to the danger of being 'kicked-out' during the training process, especially for high learning rates. The deactivation of a ReLU-neuron is irreversible.

A simple way of overcoming the kicked-off neurons, is either by restricting the learning rate or by using a slight modification the leaky ReLU activation function, which has a small positive slope for negative input values.

$$f(x) = \mathbf{I}(x < 0)(cx) + \mathbf{I}(x \geq 0)(0), \quad \text{with } c \text{ some small positive constant.}$$

2.1.3 Network architecture

More recent publications about neural networks regularly use the term 'deep learning'. Technically, deep learning and using neural networks describe the same method. Deep learning only refers to the use of multiple hidden layers within one network, while old applications often use only one hidden layer, justified by the universal approximation theorem. There is no formal definition of when a neural network is a deep network and when it is a shallow structure. State-of-art applications sometimes use up to 1000 layer networks (e.g. He et al., 2016), which is, as for today, a very deep structure. Old applications of neural networks regularly use only one hidden layer, which is unarguably shallow, since it is a minimum setup for a neural network. By many practitioners a network is called 'deep' as soon as it consists of more than one hidden layer.

As proven with the universal approximation theorem (Cybenko, 1989; Hornik, 1991), neural networks with only one hidden layer and a nonlinear activation are capable of modeling any kind of nonlinear functional form. But the theoretical capability of representing any specific form, does not mean that the 'true' function is also learnable throughout the optimization process. While neural networks are in theory universally capable of modeling any function independently of the network architecture, it cannot be guaranteed that a sufficient fit is found within a finite learning process. In practice it can also occur with a realistic chance, that the optimization process decides for an incorrect functional form, for example as consequence of overtraining (more detailed in section 2.2.4). Moreover, while it is theoretically sufficient to use a single hidden layer in order to achieve a desirable low error rate, this layer may contain a very large number of neurons, which in practice may be impossible to optimize. Deeper models, with more layers, allow to reduce the number of units per layer by increasing the representational power the same time (Goodfellow et al., 2016, p. 199).

The numerous data analytics and predictive modeling competitions especially in the field of recognition, have shown in the past decade the practical relevance and high performance of deep structures (Schmidhuber, 2015). But there is no self-fulfilling mechanism, that building deeper models increases the performance. As shown by Ba and Caruana (2014), under certain circumstances even shallow one-layer networks can perform competitively for modern applications in the field of recognition tasks. Adversely, a more recent publication by Urban et al. (2016) states that a convolutional and deep structure cannot generally be replicated by shallow structures. Therefore the matter of depth of deep learning networks is still under discussion.

2.2 Network training

Goal of network training is finding the set of model weights, which minimizes the desired error function for a given problem. The training problem consists of two components: Finding a suitable error function and then applying a suitable optimization algorithm to perform the minimization.

2.2.1 Error function

The error function $E(\cdot)$ measures the goodness of a network's fit. Goal is to find the set of network weights $\mathbf{w}$, that minimizes this function. Minimizing $E(\mathbf{w})$ is equal to minimizing the number of misclassifications and therefore maximizing the network's prediction accuracy. The cross-entropy error function aims to minimize the space between the *real* probability distribution and the distribution estimated from the data.

Following Silva et al. (2008): For $y_i = \hat{P}(c_j|\mathbf{x})$ being the estimated probability of observation i belonging to class c_j and $p_i = P(c_j|\mathbf{x})$ being the *real* probabilities coming from an underlying *true* distribution. The sample consists of $j = 1, ..., K$ different, non-overlapping classes.
Knowing the real underlying distribution, the probability of realizing the complete sample can,

assuming stochastical indepence of observations, be decomposed as

$$p(\mathbf{t}|\mathbf{x}) = p_1^{t_1} \cdot p_2^{t_2} \cdot ... \cdot p_K^{t_K}. \tag{2}$$

Accordingly for the observed, sampled probabilities

$$p_{\mathbf{w}}(\mathbf{t}|\mathbf{x}) = y_1^{t_1} \cdot y_2^{t_2} \cdot ... \cdot y_K^{t_K}, \tag{3}$$

where t_k with $k = 1$, ..., C is the number of observations assigned to each class. Of interest is the best approximation of the sample density Eq. (3) for the true density Eq. (2). This can be achieved by minimizing the Kullback-Leibler-divergence (KL). The KL-divergence measures the distance between two distributions.:

$$\begin{aligned}
\mathrm{KL}\left(\frac{p(\mathbf{t}_i|\mathbf{x}_i)}{p_w(\mathbf{t}_i|\mathbf{x}_i)}\right) &= \sum_{i=1}^{N} \log\left(\frac{p(\mathbf{t}_i|\mathbf{x}_i)}{p_{\mathbf{w}}(\mathbf{t}_i|\mathbf{x}_i)}\right) &(4)\\
&= \sum_{i=1}^{N} \log\left(\frac{p_{1,i}^{t_{1,i}} \cdot p_{2,i}^{t_{2,i}} \cdot ... \cdot p_{C,i}^{t_{C,i}}}{y_{1,i}^{t_{1,i}} \cdot y_{2,i}^{t_{2,i}} \cdot ... \cdot y_{C,i}^{t_{C,i}}}\right) &(5)\\
&= -\sum_{i=1}^{N}\sum_{k=1}^{C} t_{k,i}\log(y_{k,i}) + \sum_{i=1}^{N}\sum_{k=1}^{C} t_{k,i}\log(p_{k,i}) &(6)
\end{aligned}$$

The true model probabilities $p_{k,i}$ are unknown, but independent of the model weights $\mathbf{w}$. Therefore it is equivalent to minimizing

$$E_{CE}(\mathbf{w}) = -\sum_{i=1}^{N}\sum_{k=1}^{C} t_{k,i}\log(y_{k,i}) \tag{7}$$

The binary cross-entropy is a special case of the multiclass cross-entropy, where the number of classes is $C = 2$:

$$E_{CE}(\mathbf{w}) = -\sum_{i=1}^{N} t_i \log(y_i) + (1 - t_i)\log(1 - y_i))$$

2.2.2 Optimization methods

The success of deep learning in the past decade is because of overcoming two major bottlenecks in network training: advances in optimization algorithms and the vast supply of inexpensive computational power. The following section discusses different gradient based optimization methods.

Gradient descent Very early deep learning publications (e.g. Rumelhart et al., 1985) already consider gradient descent (GD) training as optimization procedure. The (batch) GD-optimizer is an iterative method, which updates the network weights $\mathbf{w}$ based on the gradient of the error function:

$$\mathbf{w}^{(\tau+\mathbf{1})} = \mathbf{w} - \gamma \nabla E(\mathbf{w}^{(\tau)}) \tag{8}$$

With each iteration τ the model weights are moved into the direction of the gradient and the process is repeated for the complete training data (the whole *batch*). The parameter $\gamma > 0$ is called the learning rate, which fixes the step size for the weight updates. The choice of γ is crucial: If γ is too small, the convergence may take very long. If γ is too large, it might be that no convergence is achieved, because the algorithm cannot settle in the local or global minimum of the error function E($\mathbf{w}$) (Duda et al., 2001, p. 225). Another shortfall of the gradient descent method is that for large datasets each iteration may be very computational intense with many redundant calculation steps.

Stochastic gradient descent (SGD) The on-line version of the gradient descent method, called the stochastic or incremental gradient descent is of more practical use for larger datasets (Bishop, 2006; LeCun et al., 1989). The overall error-function for a set of independent observations is the sum of the individual errors for each data point.

$$E(\mathbf{w}) = \sum_{i=1}^{N} E_i(\mathbf{w})$$

SGD updates the weights after considering only one single observation at a time. The weight update is then repeated according to (8). The new observation for the next update step needs to be chosen randomly from the dataset, otherwise the algorithm also learns from the arrangement of the input data. The stochastic gradient descent is usually unsteady and jumps a lot in the convergence process, since the weight updates have a high variance. This can also be an advantage, since the random behavior allows to overcome local minima.

It is also possible to use any kind of intermediate procedure between updating the weights after one observation or after the whole training data. Those procedures are called mini-batch SGD and are commonly used, because they combine the benefits of both extremes. The parameter updates have a lower variance than in SGD, it is more efficient than GD and still less likely to get stuck in local extremes. Dependent on training set size and specific problem, the common choice for the mini-batch size lies in practice between 50 and 250 (Ruder, 2016).

Adam-optimizer Adam-optimizer by Kingma and Ba (2015) is a state-of-art gradient based optimizer. Ruder (2016) compares and empirical tests different optimization methods and choses Adam to be the method of choice, due to its favorable behavior in practice.

Adam (Adaptive Moment Estimation) computes adaptive learning rates for each parameter. Additionally, the decaying average of past squared gradients, and the average of past gradients are considered in the optimizer, combining advantages of other advanced optimizers as Adadelta (Zeiler, 2012) and the momentum method (Qian, 1999). The Adam-optimizer uses for the parameter update information, gathered from the first and also second moment. Certain hyperparameters are required for the optimization process:

- γ : learning rate, step-size for weight update
- β_1: exponential decay for first moment, $\beta_1 \in [0, 1)$
- β_2: exponential decay for second moment, $\beta_2 \in [0, 1)$
- ϵ: small positive constant to prevent division by zero

The procedure of the Adam-optimizer is described by Kingma and Ba (2015) with the a pseudocode in Fig. 4. For each iteration, the algorithm considers a different, random mini-batch of the dataset.

Algorithm 1 Adam-optimizer

Require: γ, β_1, β_2, ϵ hyperparameters
Require: $f(\theta)$ stochastic objective function with parameters θ
Require: θ_0 initial parameter vector
$m_0 \leftarrow 0$ initialize first moment vector
$v_0 \leftarrow 0$ initialize second moment vector
$t \leftarrow 0$ initialize time step
while θ_t not converged **do**
  $t \leftarrow t + 1$
  $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ calculate gradients
  $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ update 1st moment (biased)
  $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ update 2nd moment (biased)
  $\hat{m}_t \leftarrow m_t \,/\, (1 - \beta_1^t)$ bias correction 1st moment
  $\hat{v}_t \leftarrow v_t \,/\, (1 - \beta_2^t)$ bias correction 2nd moment
  $\theta_t \leftarrow \theta_{t-1} - \gamma \cdot \hat{m}_t \,/\, (\sqrt{\hat{v}_t} + \epsilon)$ parameter update
end while
return θ_t

Figure 4: Pseudo code Adam-optimizer, own representation based on Kingma and Ba (2015)

Default values for the parameters, which show a good behavior in practice are given by Kingma and Ba (2015) as $\gamma = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$.

2.2.3 Back-propagation

To produce the desired estimation output $\hat{y}$ the information is passed strictly forward through the neural network, which can be understood as a complex chain of single operations. This stream, starting at the input layer passing through the hidden units until it ends in the calculation of the network's error function, is called forward propagation. The optimization procedures described in section 2.2.2 are gradient based methods, meaning that they take information from the error function's gradient to update the model weights. The process of calculating the gradient is called back-propagation, since, in analogy to forward-propagation, the gradient of the error function flows backward through the network. (Error) back-propagation can be understood as applying the chain rule of calculus several times at every node of the network. That way, the gradient of very complex structures, like deep neural networks, is more feasible to calculate as a number of small sub-problems.

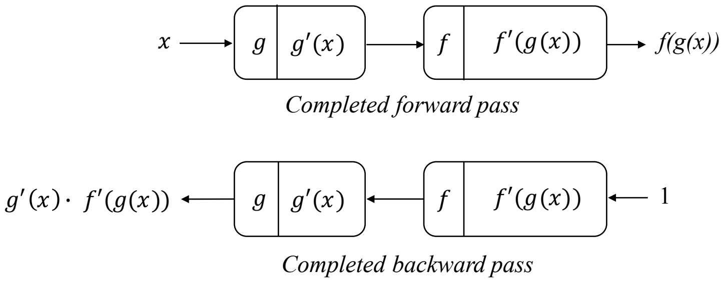


Figure 5: Example of back-propagation, with arbitrary activation functions g(·), f(·). Result of back-propagation equals chain-rule of calculus. Own representation based on Rojas (1996, p. 159)

According to Goodfellow et al. (2016) back-propagation is often misunterstood to be the whole learning algorithm for artificial neural networks, but in fact it is only the method for computing the gradient. The actual learning is perfomed by using a optimization algorithm as described in section 2.2.2. The numerical evaluation of the gradient is one of the main computational problems in network training, even if it might be analytically rather simple. While the method itself is much older and not limited to neural networks, efficient error back-propagation in the context of multilayer perceptrons with gradient descent was made popular by Rumelhart et al. (1985).

Back-propagation can best be understood when thinking of a neural network as a computational graph. Computational graphs break down mathematical expressions in a series of subordinate problems that can be treated independently. In the back-propagation process, the gradient for every point of the network can then be determined by applying the chain rule of calculus (for proof, see e.g. Rojas, 1996).

Each neuron is technically functioning in two directions: in the forward pass it applies the nonlinear activation function, in the backward pass the first derivative of the activation function is used. In the forward pass, the derivative on the input of this specific node is also calculated and stored. The backward pass only calculates the gradient of the whole network, which is done by applying the chain rule of calculus. Assuming a very simple case, of a two layer network with one node in each layer, see Fig. 5, for illustration.

2.2.4 Challenges in network training

Overfitting As stated by the universal approximation theorem (see chapter 2.1.2) any kind of functional form can be modeled by a neural network of sufficient size. In practice the model training process is often tackled by the problem of limited availability of training data. For deep networks with large layers this leads to the problem, that the network can be highly affected by sampling noise. Consequently the model is over-adapted to the dataset, having high capabilities in describing the given observations, but shows poor results in the out-of-sample prediction for new unseen data. This problem is called overfitting or overtraining (see for more detail Tetko et al., 1995). The most intuitive way to reduce overfitting is increasing the number of training data, but depending on the specific application of the network this is not always a feasible approach.

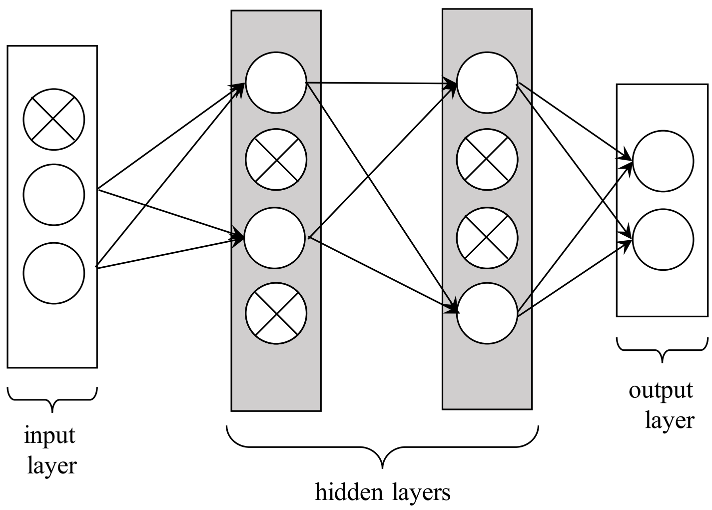


Figure 6: Network after after applying dropout: Crossed out nodes are deactivated.

Dropout One method brought up by Hinton et al. (2012) is adding dropout to the network layers. Dropout layers randomly deactivate neurons along with their connections during training time (Fig. 6). This prevents learning too specific structures, which are only a property of the sampled data, but not of the underlying true process. It reduces the number of overall neurons in the network from n to np neurons, where p is the sampling probability ($0 < p < 1$). That way several random networks with a thinner structure are estimated. For the testing process, the overall effect of each node can be calculated by averaging over the weights, yielding again a full network with n nodes. Compared to other regularization methods, dropout shows an improved performance when it comes to computational efficiency and predicitive power on many benchmark datasets for machine learning problems (Srivastava et al., 2014).

Regularization Other than randomizing the elements of a network, the overfitting problem can be reduced by penalizing the size of the model. Bigger models have in general a better ability to model functional relations, but they are more prone to model spurious relations. Regularization methods penalize bigger network sizes by adding an additional parameter to the error function.

Possible methods are the L1- or L2-regularization method, where the latter is more commonly used in practice (Nielsen, 2015). The L2- or weight decay regularization extends the error function by the penalty factor $\frac{\lambda}{2N}\sum_w w^2$. This equals the sum of all squared adjustable weights throughout the network, without the biases, scaled by a factor, which is dependent on a constant λ and the sample size N. The size of λ determines the strength of penalization for model size and weight size. The parameter λ is a hyperparamter free of choice, depending on the specific dataset and the network target. The smaller λ, the closer the regularization tends to the unrestricted error function.

Small model weights are desirable, because the network's classification decision should not only depend on a few specific features. It should rather take a variation of features with lower weights into consideration, which is supposedly more robust to out-of-sample predictions and less dependent on the seen training data.

In the case of categorial cross-entropy from Eq. (6), the optimization problem changes to

$$E_{CE}(\mathbf{w}) = -\sum_{i=1}^{N}\sum_{k=1}^{C} t_{k,i}\log(y_{k,i}) + \frac{\lambda}{2N}\sum_{w} w^2.$$

The L1-regularization, or elastic net regulization, adds the term $\frac{\lambda}{n}\sum_w |w|$ to the error function. Despite having a similar intuition of penalizing the network size and the size of weights, the L1-regularization shows differences in behavior: While the L2-regularization leads to a weight vector, that has diffused, small weights, L1-regularization produces vectors, which are based on the main features, pushing all other weights to zero and consequently making the weight vector sparse. Unless a precise feature selection is the target, L2-regularization leads generally to better results (Nielsen, 2015).

2.3 Image classification with convolutional neural networks

The ambition of this thesis is building a neural network, that classifies pictures into a fixed number of classes. The before discussed methods are general bricks for building neural networks and can be used for any task. For most problems, the data is fed into the network by an one-dimensional vector of numbers. To transform a picture as the human eye can see it into a machine readable input, the colors for each pixel are converted into a binary representation. Pictures are arranged in three dimensions: width, height and depth. Width and height refer to the number of pixels in the first two dimensions. The depth is the dimension of the color scale.

Pure greyscale pictures have a depth of one and are converted using a color table, where black is represented by the number 0, white is represented by 255. This holds for 8 bit per pixel color depth tables ($2^8 = 256$), which are commonly used for this task. If the picture is colored, the red, green and blue (RGB) spectrals of the picture are treated separately, giving a representational matrix with depth three. For each pixel and each of the RGB-layers, the possible values range again between 0 and 255.

The standard version of a network as in Fig. 1, where every neuron is connected to the subsequent layers, is called a fully connected layer. For different purposes, and data structures, different layer designs can be chosen. In a fully connected design, the raw data is given into a neural network by one-dimensional vector of. The nature of pictures is two-dimensional and grid-like (pixel height and width). It is possible to convert the two-dimensional picture into a one-dimensional vector by simply stacking the rows (or columns), but that takes out significant information of the raw data.

Convolutional Networks use among the already mentioned fully-connected layers two special kinds of layers: *convolutional layers* and *pooling layers*. The idea of convolution in neural networks, initially called 'neocognitron', was first mentioned by Fukushima (1980), inspired from a model by Hubel and Wiesel (1962) about simple and complex cells in cats' visual primary cortex. Simple cells pass on a signal when detecting simple patterns, complex cells are rather invariant to spatial variations (Schmidhuber, 2015). Simple cells are an analogy to convolutional layers, complex cells have a similar function like pooling layers.

Since 2010 the training of CNNs using GPUs is standard, which allows a faster training and therefore wider applicability of CNN technology (Krizhevsky et al., 2012). CNNs are component of almost all winners of international image classification competitions, such as the ‘imagenet large scale visual recognition challenge’, which is often referred to as the benchmark of state-of-art image classification.

2.3.1 Convolutional layers

The motivation for using convolutional layers is that the 2D-structure of a picture shall be exploited for information processing. Intuitively, pixels that are close to each other in an image, have a higher relevance for recognition of the complete object when they are spatially close. A specific structure that is built up by a number pixels is called a feature. After determining those local features on a small excerpt of the pictures, they can be merged on higher-level layers until finally the whole picture is identifiable. Thereby the network goes from very generic forms and patterns that are almost independent of the network’s scope to features in later layers, which are highly dependent on the used picture and objective of the dataset.

Convolutional layers are characterized by the following hyperparameters:

- the filter size (F), or receptive field, for which usually a size of 3×3, 5×5 or 7×7 is chosen,
- the stride (S), by which the filter is moved along the feature map (integer value ≥ 1),
- zero-padding (P) (optional), which allows to control the output volume size, by padding the input volume with zeros,
- the filter depth (D) (not to be confused with the color-depth), the number of filters, which are used for each convolutional layer.

For a fixed size of square input-volume $V \times V$, which is in the input layer determined by the size in pixels of the picture, the output size after the first convolution has the dimensions:

$$A \times A \times D, \quad \text{with} \quad A = \frac{V - F + 2P}{S} + 1. \tag{9}$$

It is necessary to chose the parameters F, S, P in a way, that A results in a integer value. The filter depth D can be chosen freely.

Weight sharing Besides the spatial considerations of the input matrix, convolutional layers impose a special restriction on the number of adjustable weights. Unlike in fully connected layers, each filter uses the same weights throughout the complete picture, leading to a flashlight-like structure. This procedure is based on the assumption, that if one filter extracts a useful feature at some position of the picture, this filter should also be useful at other positions. This proves also in practice to be true, since structures found by filters are almost universally generic, looking for distinct edges and lines (Krizhevsky et al., 2012).

Considering the following example, to illustrate the effect on the number of weights: The input volume V has dimensions $[48 \times 48 \times 1]$, last channel being the color depth, after a first convolution with $F = 5$, $S = 3$ and $P = 1$ and a convolutional filter depth of $D = 64$. The output volume of the convolution has according to Eq. (9) of $15 \times 15 \times 64 = 14400$ single neurons, of which each has $[5 \times 5 \times 1]$ weights. If no weight sharing is applied, only the first convolutional layer has $360,000$ weights (plus biases), that need to be optimized throughout the training process, rising with every layer. For very deep structures, as they are common for modern convolutional networks, this number can become extremely high. By applying weight sharing, the weights are the same for each layer of D, which reduces the number of parameters to $[64 \times 5 \times 5 \times 1] = 1600$ plus the number of biases.

Parameter sharing also causes equivariance to translations within the convolutional layers. Equivariance holds for e.g. shifts of the input, other transformations such as rotation or scaling are not equivariant to the convolutional function (Goodfellow et al., 2016, p. 339).

2.3.2 Pooling layer

The desired classification output is usually much smaller in dimension than the size of the input volume V. Pooling layers achieve stepwise reduction of the matrix dimension in-between the convolutional layers. This also reduces the computational intensity of the network. Among other choices one common option is *max pooling*. The input volume is scanned depth-layer-wise with some spatial extent F, applying the max-function. As for the convolutional layers, stride S can be chosen freely under the constraint that the resulting dimensions for the output volume are integers. The calculation for the output size is analogously as in (9), the filter-depth D remains unchanged throughout the operation. Commonly $F = 2$ and $S = 2$ are applied, or $F = 3$ and $S = 2$, which leads to a overlap in the pooling procedure and reduces the probability of overfit (Krizhevsky et al., 2012). Larger spatial extent F of the receptive field in the pooling layer are acting more restrictively on the model capabilities. Pooling layers make the representation approximately invariant for small input translations. This is a desirable property for image processing, since for the majority of features it is of bigger interest, whether the feature is present in the image and not its precise position (Goodfellow et al., 2016, p. 342)

2.3.3 Network fine-tuning

The specific design of a (convolutional) neural network is consisting of numerous free-of-choice parameters and hyperparameters. Applying grid search to determine the optimal set of hyperparameters and the optimal design is reasonable, but often not feasible, due to the computational complexity. A different approach is applying transfer learning. Transfer learning takes knowledge already gathered in one task and applying it to another target. It is therefore intuitively very close to the human learning process, which is largely based on previous experiences, that can widely differ in circumstances. Transfer learning can help in three ways: First, the model starts off with a higher initial knowledge, making better predictions right from the beginning, second the learning time can be decreased, third the final level of accuracy is higher than without learning (Torrey and Shavlik, 2009). This is achieved by training the base network with its original target, then fixing the first n layers and only training the classifier on top. A slight variation is fine-tuning, where the errors and weights are not fixed during back-propagation, but are also adjusted to the new learning objective. Which method to use depends on the size and kind of the data. For a small dataset and a large base model, there is significant risk of overfitting when fine-tuning, therefore 'freezing' the layers in the optimization process is preferred (Yosinski et al., 2014).

When looking at a neural network, features need to go from very general in the lower levels of the network, to highly specific for the learning task, in the last layers of the network. It is found, that the generic low level filters in convolutional layers, are almost independent of the training objective and further model specifications (Yosinski et al., 2014; Krizhevsky et al., 2012) and therefore can also be used for other tasks.

3 Facial expression recognition

Ambition of this thesis is to measure nonverbal communication and evaluate its effect on financial data during live events. The quantification of nonverbal communication will be achieved by constructing a deep convolutional neural network, which is able to classify facial expressions into several emotions. In this sections, different model designs are discussed, trained and evaluated on a suitable dataset.

3.1 Deep neural networks

In this subsection a deep neural network is built and trained, which has the purpose of recognizing facial expressions. Deep neural networks need a large amount of classified data to adjust the weights during training process to fit a specific problem. The here used dataset is taken from the 'Kaggle' competition "Challenges in Representation Learning: Facial Expression Recognition Challenge". Kaggle is a platform for machine learning competitions. Data for competitions is provided by companies or researchers.

The Facial Expression Recognition 2013 (FER2013) dataset is created by Pierre Luc Carrier and Aaron Courville, using the Google image search API with respective keywords. For each picture a face-rectangle is set automatically by a software. Each label and face-rectangle is approved by a human supervisor, rejecting misspecified pictures. Each picture is cropped along the face-rectangle and converted into a grayscale square-cut shape of 48x48 pixels (see Goodfellow et al. (2015) for more information on dataset generation). Based on some small-scale experiment, Goodfellow et al. (2015) determine a $65 \pm 5\%$ accuracy for humans to solve this classification task on the given dataset.

The final FER2013 dataset contains 35887 images with the following emotions, count of appearance in the dataset, and their original encoding within the dataset:

- 0 = anger (4953 images)
- 1 = disgust (547 images)
- 2 = fear (5121 images)
- 3 = happiness (8989 images)
- 4 = sadness (6077 images)

- 5 = surprise (4002 images)
- 6 = neutral (6198 images)

The dataset is oversampled with the class 'happiness', while 'disgust' is underrepresented. The other classes are roughly evenly sized. FER2013 contains a wide variability of ages, looks and people. Fig. 7 shows a random sample of example images. Some of the pictures seemingly show acted emotions, others appear to be real snapshots. Also faces from paintings are in the dataset. Some pictures show occlusions like glasses, hats or hands covering parts of their faces, or watermarks by the right-owner of the picture. The head poses vary from frontal portrait pictures to side-face shots, not all adjusted along a vertical axis. Moreover the illumination of the single images varies heavily throughout the dataset.

3.1.1 Data preparation

The original FER2013 dataset consists of a .csv-file, containing the vectorized 8-bit grayscale versions of each image and the corresponding label. Each pixel vector has the dimension of 48*48 (= 2304). One row represents one picture. For using any kind of neural networks, it is common to standardize the values of the input to a range from 0 to 1. Since the input matrix only has values in the interval between 0 and 255, this is achieved by dividing each value within the input matrix by 256. The standardization in the context of picture manipulation, represents an assimilation of illumination. In deep learning the input matrix is often also normalized by dividing each element by the standard deviation and zero-centered by subtracting the mean. Normalization is not done as data preparation step, but will be repeatedly performed within several batch normalization layers throughout the network.

Convolutional neural networks are specifically made for processing the natural 2D-grid structure of pixels. Therefore it is necessary to arrange each vector from dimension 1×2304 to a matrix with dimensions $48 \times 48 \times 1$. The last dimension equals the color scale, which is grayscale in this dataset. Throughout this paper, the colors-last notation is used, which is the standard notation in the 'Theano'-library for Python.

The emotion-label is encoded by integers from zero to six (0 - anger, 1 - disgust , 2 - fear, 3 - happiness, 4 - sadness, 5 - surprise, 6 - neutral). When using categorical cross-entropy as error metric for training, it is necessary to define the label in a one-of-K-scheme or one-hot encoding. The one-hot encoding represents each label as a series with a single high, 1, while the rest of the

Figure 7: Random examples from FER2013 dataset. One emotion per row, from top to bottom: anger, disgust, fear, happiness, sadness, surprise, neutral

classes are represented by zeros. For example: For a label of class 2 (fear), the one-hot encoding is 0010000; for a label of class to 6 (neutral), the one-hot encoding is 0000001, etc.

3.1.2 Convolutional neural network

This subsection discusses different models options under two different approaches: First, a model following a general design for image classification. Hyperparameters are chosen based on a literature overview on the topic of facial expression recognition, model weights are trained from random initialization using a common optimization methods. Second, a fine-tuning approach based on the VGG-face network is applied for the FER2013 data. The VGG-face network is a deep neural network structure with the purpose of general face recognition. Fine-tuning is a method to transfer knowledge of the model weights to a new dataset The resulting model quality will be compared to Microsoft Emotion API, a Microsoft service with the same task.

Model Architecture The problem of facial expression recognition has been focused upon in research with different methods (Tian et al., 2011; Manglik et al., 2004; Lawrence et al., 1997). The following analysis will focus on estimation using one single convolutional neural network. Pramerdorfer and Kampel (2016) compare different state-of-art-procedures for facial expression recognition. Based on their research the model design developed by Yu and Zhang (2015) is implemented, because it combines a simple structure and a good performance. The chosen network is a 7 hidden-layer CNN, displayed in Tab. 1. Determining the depth of a network, only the layers with adjustable weights are counted, in this case the convolutional layers and the fully connected (dense) layers are counted. Layers such as the input layer, max-pooling, dropout, or flattening layers have fixed weights. The model is generated and trained using the Keras package for Python (Chollet and Others, 2015), using a Theano (Al-Rfou et al., 2016) backend. Keras is a modular and fast-to-implement API especially for deep neural networks. Theano performs efficiently the numerical computation.

The size of the input layer is determined by the shape of the input images. For the FER2013 data, this is fixed to a 48×48 pixel size, with one color channel (depth one), since all pictures are grayscale. The size of the the convolutional filters is 3×3. The receptive field moves with a stride of one. Larger strides reduces the size of the input matrix faster, but since 48×48 pixels has to be considered as already small sized input matrix, a too fast reduction is undesirable. The small stride also ensures that the representation is invariant to small input translations. The first two convolutional layers have a depth of 64, increasing to 128 for later convolutional layers of the network. Each layer of the filter depth is trained to detect a certain feature within

	layer type	output dimension	filter	stride	# parameters
1	input	48 × 48 × 1	3 × 3	-	0
2	convolution	46 × 46 × 64	3 × 3	1	640
3	convolution	44 × 44 × 64	3 × 3	1	36,928
4	max-pooling	22 × 22 × 64	2 × 2	2	0
5	convolution	20 × 20 × 128	3 × 3	1	73,856
6	convolution	18 × 18 × 128	3 × 3	1	147,584
7	max-pooling	9 × 9 × 128	2 × 2	2	0
8	flatten	1 × 10,638	-	-	0
9	dropout	1 × 10,638	-	-	0
10	dense	1 × 1024	-	-	10,617,856
11	dropout	1 × 1024	-	-	0
12	dense	1 × 1024	-	-	1,049,600
13	dense	1 × 7	-	-	7,175
Σ					$11,933,639$

Table 1: Specification CNN for FER2013 data set FVCConvNet

the picture. Rectified linear units are chosen as activation functions, the final class prediction is achieved by using softmax-activation.

The trained convolutional neural network consists of 11,933,639 trainable parameters, which summarizes all weights and biases in the model. Tab. 1 shows the number of parameters per layer. The low number of parameters in the lower levels of the network, compared to the fully connected layers, are due to the weight sharing restriction in convolutional layers.

The pooling layers use the max-function with a receptive field of 2×2 and a stride of 2, which reduces the input size according to Eq. (9) approximately by half with each pooling layer. Due to the small format of the input format, pooling layers are used every two convolutional layers. Each pooling layer is followed by a batch-normalization layer (Ioffe and Szegedy, 2015), which speeds up computation and estimation efficiency, since zero centering is a desired property to reduce the risk of overfitting.

After the convolution and pooling procedures, the neurons are flattened from their 2D-structure into a vector. Afterwards two fully connected layers are added, in which every neuron between the two layers are connected, using a 20 % dropout to reduce the risk of learning structures, which are too sample-specific.

For optimization the Adam-algorithm is applied, a version of stochastic gradient optimization. The parameter values and a learning rate of 0.001 are chosen as proposed in the original paper by Kingma and Ba (2015). The initial values for the model weights are randomly chosen. Along with the convention of neural networks and the training dataset size, a batch size of 125 is used. The optimization process is repeated over 15 iterations. Categorical cross-entropy determines the accuracy or loss of each prediction.

Many different specifications of model architecture and hyperparameter choice were tried, but with either no effect or a worsening on the model's predictive power.

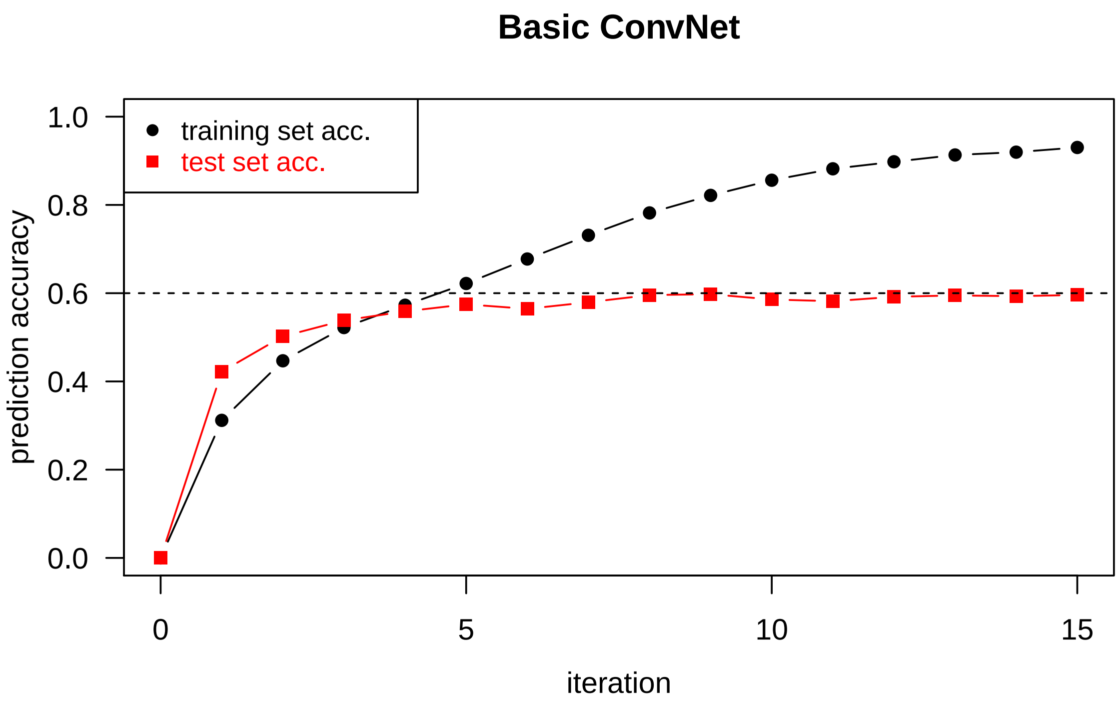


Figure 8: Learning curve of convolutional neural network in Tab. 1 FVCperformance

Estimation results Fig. 8 shows the prediction accuracy for each of the 15 repetitions. The model rises to 40% accuracy already after the first epoch. Within 5 iterations it reaches its maximum prediction power of 60% on the testing data and remains on this level, meaning that the model classifies 60% of never seen images correctly into one over seven existing classes. On the other hand, the accuracy within the training set, which is used to optimize the weights, the accuracy converges with a growing number of iterations against 100%.

Fig. 8 allows to evaluate the choice of the learning rate, which seems to be in an accurate range. The accuracy rising up to almost 100% is a good indication for sufficient model depth for this specific problem. The network has the highest precision in recognizing the emotions happy, neutral and disgusted as can be seen in Fig. 9. The numbers on the main diagonal in Fig. 9 have to be understood as positive predictive value (PPV)

$$PPV = \frac{TP}{(TP + FP)}. \tag{10}$$

PPV equals the ratio of true positives (TP) (observation i from class k is predicted as class member of k) over true positives plus false positives (FP) (observation i from class $l \neq k$ is predicted to belong to class k), which sums up to all observations that were predicted as members of class k. This resembles the model's precision per class. The off-diagonal shows consequently the false-recovery rate as the proportion of misclassified observations over all predictions per class.

Yu and Zhang (2015) use a dropout-layer before starting the convolution, which is left out in the here implemented model, since it has shown to be a bottleneck in the model's prediction capacity. Without droppput as first stage an increase in accuracy by approximately 10% can be achieved. This result may be reasonable, since the first stage of dropout is a harsh cropping of the picture and reduces size by half. The original model also uses stochastic-pooling layers, here max-pooling layers are used, since stochastic-pooling is not implemented in Keras.

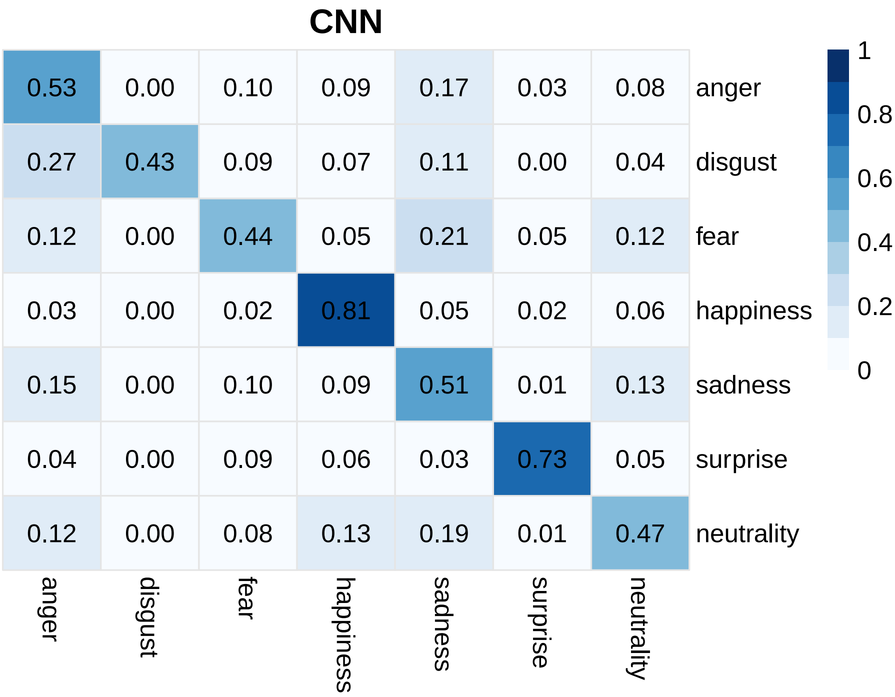


Figure 9: Prediction Quality of CNN. On-diagonal: positive predictive value, off-diagonal: false discovery rate FVCheatmap

3.1.3 Fine-tuned neural network

Model architecture Next to training a neural network from zero, the fine-tuning approach is popular when applying deep learning in practice. Convolutional neural networks show a generic behavior on the low levels of a network, in which the specific dataset is only of minor importance, and become more specific with the depth of the network.

Fine-tuning uses a very deep and precisely trained network on generic data and only adjusts the last specific layers to the new dataset. Some of those very deep networks, including their weights, which are trained over long periods on multi-GPU computers, are published by deep learning enthusiasts and researchers for public use. Even though the specific target of the base model is not of the major importance, a very deep network is used here, which was specifically trained for processing faces. Model description and weights are published for non-commercial use by the Visual Geometry Group (VGG) of the University of Oxford. The VGG-face CNN (Parkhi et al., 2015) is based on a very deep structure from VGG16, and its weights are trained specifically on faces (Huang et al., 2007; Wolf et al., 2011).

The background for the VGG-face project is, that labeled facial datasets, as needed for training a deep networks, are sparse for free public use. The same time big commercial companies such as Facebook or Google are already in possession of large data varieties, which are restricted for public due to privacy restrictions and also serve as a competitive resource. Parkhi et al. (2015) combine different datasets to achieve a large facial dataset of over 980 000 unique pictures.

A 16-layer-CNN is set up for the face recognition problem, which can be considered as a very deep structure. The main parts of the VGG-face-CNN are the same as in the model built in section 3.1.2: Convolutional filters with a filter size of 3×3, rectified linear units as nonlinear acitvation functions and a small stride size of only 1, which guarantees a high level of feature invariance. Convolutional layers are followed by maxpooling layers. The model ends with a set of fully connected layers, the last layer uses a softmax activation to predict the class probabilities.

For fine-tuning this network to our specific question, the convolutional layers are kept, and then two fully connected layer with 512 hidden units are adjusted to the FER dataset, using rectified linear units as activation functions and a softmax layer for class prediction. It has to be considered that the original data for the VGG-face model has dimensions $224 \times 224 \times 3$, while a single observation in the FER-dataset has dimensions $48 \times 48 \times 1$. The first two dimensions can be adapted by zero padding, the third dimension is the color-depth. While the FER dataset is grayscale and has color depth one, the VGG-face dataset is colored and has therefore depth three (red - green - blue). The color depth of three can be simulated by simply concatenating the identical matrix three times in the third dimension. This imitates the red-green-blue representation of a grayscale picture: all three channels are identical.

Estimation results Despite being a state-of-art procedure, applying a deep structure to classification problems with limited data and a different structure, is problematic. The fine-tuning approach shows significant worse results in the predictive power than the neural network, with a more shallow structure and a random initialization of parameters in section 3.1.2.

After the first iteration, the model performs significantly worse than the basic CNN. The model's convergence is overall slower and seems to get stuck on a plateau in the beginning. After 10 iterations the final predictive power of approximately 52% is reached (Fig. 10) and remains for the last 5 iterations. This is almost 10% less than the competitive model.

The model analysis based on the confusion matrix (Fig. 11) exhibits that the model has severe prediction problems. The prediction accuracy is only for 'happiness' and 'surprise' satisfactory. Generally, the fine-tuned VGG-face model overpredicts neutrality as the dominant emotion,

	layer type	output dimension	filter	stride	# parameters
1	input	48 × 48 × 3	1×1	-	0
2	convolution	48 × 48 × 64	1×1	1	1,792
3	convolution	48 × 48 × 64	1×1	1	36,928
4	max-pooling	24 × 24 × 64	2×2	2	0
5	convolution	24 × 24 × 128	1×1	1	73,856
6	convolution	24 × 24 × 128	1×1	1	147,584
7	max-pooling	12 × 12 × 256	2×2	2	0
8	convolution	12 × 12 × 512	1×1	1	295,168
9	convolution	12 × 12 × 512	1×1	1	590,080
10	convolution	12 × 12 × 512	1×1	1	590,080
11	max-pooling	6 × 6 × 512	2×2	2	0
12	convolution	6 × 6 × 512	1×1	1	1,180,160
13	convolution	6 × 6 × 512	1×1	1	2,359,808
14	convolution	6 × 6 × 512	1×1	1	2,359,808
15	max-pooling	3 × 3 × 512	2×2	2	0
16	convolution	3 × 3 × 512	1×1	1	2,359,808
17	convolution	3 × 3 × 512	1×1	1	2,359,808
18	convolution	3 × 3 × 512	1×1	1	2,359,808
19	max-pooling	1 × 1 × 512	2×2	2	0
20	flatten	1 × 512	-	-	0
21	dense	1 × 512	-	-	262,656
22	dense	1 × 512	-	-	262,656
23	dense	1 × 7	-	-	3,591
					Σ 15,243,591

Table 2: Specification VGG16 model for FER2013 dataset FVCfinetuning

while not being able to detect other emotions. It can be followed, that the fine-tuning method could not adjust well to the given task.

It is possible, that the model would gain significant predictive power if it would be trained for a longer period. Also the model's training set accuracy remains at around 70%, which may be an indication that the model would need more iterations to improve its performance. Another reason might be a suboptimal choice of the learning rate, which may cause the plateau behavior in the beginning.

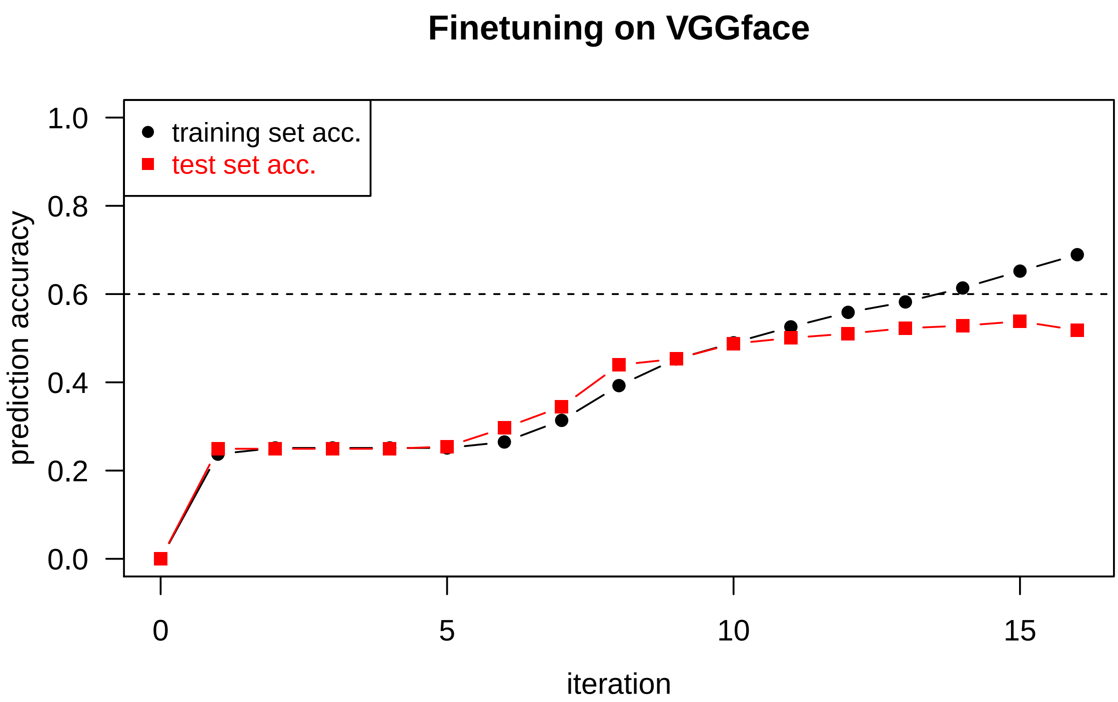


Figure 10: Learning curve convolutional neural network in Tab. 2, horizontal line: reference performance model Tab. 1 FVCperformance

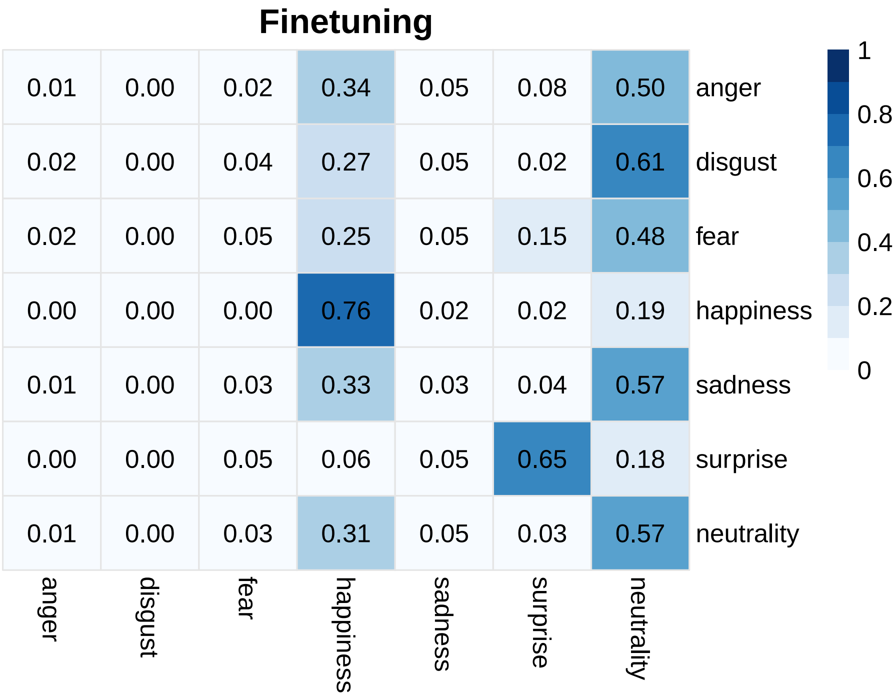


Figure 11: Prediction Quality of fine-tuned VGG-model. On-diagonal: positive predictive value, off-diagonal: false discovery rate FVCheatmap

3.1.4 Comparison with Microsoft Cognitive Service

The following subsection compares the performance of Microsoft Emotion API and the previously discussed models based on FER2013 dataset. Microsoft Cognitive is a service by Microsoft Corporation, which uses machine learning algorithms to offer easily accessible and integratable artificial intelligence services, such as speech recognition, sentiment analysis, face recognition and also facial expression recognition. The Emotion API is the service for emotion classification of pictures. For this paper Python is used to access the API (FVCcall_API). The Microsoft service is a black-box-model and no detailed information is offered about the behind lying machine learning technology.

The emotion-labels of the Emotion API and the FER2013 dataset differ slightly. The Emotion API from Microsoft is able to classify eight different facial expressions: anger, contempt, disgust, fear, happiness, neutrality, sadness and surprise. The set of seven emotions from FER2013 is extended by an additional emotion (contempt).

The labeled test data from the previous models is used to determine the network's predictive power. The evaluation is made based on the testing fraction of FER2013. A first comparison of the labeled data from the original dataset versus the predicted emotions yields an accuracy

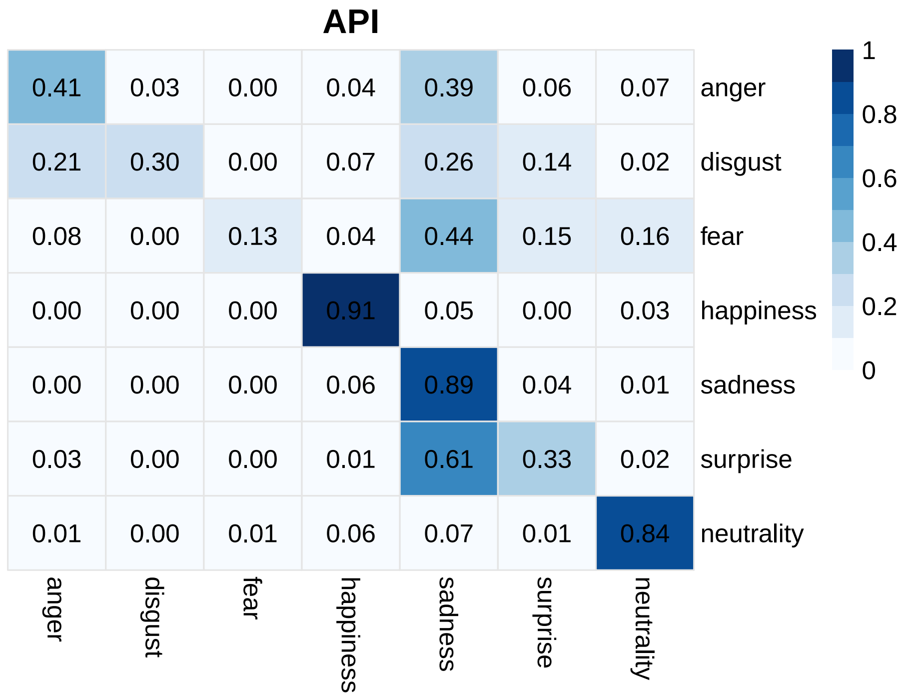


Figure 12: Prediction Quality of Microsoft Emotion API. On-diagonal: positive predictive value, off-diagonal: false discovery rate FVCheatmap

of 41%, which is compared to the previous models significantly worse. Investigating the reasons for this bad accuracy shows that the Emotion API is unable to detect a face-rectangular for 35% of the requested pictures. This is due to occlusions like hats, watermarks, or glasses, invalid clipping or just inaccurate pictures. Since the size of each picture is with 48×48 pixels already small compared to real world applications, the pictures become uninterpretable. This could also explain the inferior results of the fine-tuning method using VGGface as base model in section 3.1.3.The problem of the different number of classes can be neglected for the further analysis, because out of over 2300 pictures only 17 faces are classified as showing ‘contempt’.

The test dataset is cleaned from pictures which do not have interpretable faces for the Emotion API and the process is rerun. The fine-tuned model from section 3.1.3 shows now an accuracy of slightly above 56% for the smaller dataset, which is an improvement of 4%. The basic CNN model with random parameter initialization from section 3.1.2 improves its accuracy from 60% to 63%. As expected, the Emotion API improves significantly by 22% to 63 % for the pictures with detectable faces. It can be concluded that the Emotion API and the basic CNN model with random parameter initialization from section 3.1.2 are competitive in their prediction accuracy.

3.2 Discussion of results

The human accuracy for solving this task is according to Goodfellow et al. (2015) $65 \pm 5\%$. This interpretation of emotions from a static image can also for humans be a complicated task. The interpretation of facial expressions is often based on the background and situation. Moreover, the number of seven emotion classes is not a final selection and also many emotions could be interpreted as belonging to more than one class, which is assumed to be impossible in this setting. Also misclassifications in the dataset and bad data quality can not be fully excluded, as some pictures may simple be uninterpretable. This hypothesis is also backed, when using the Microsoft Emotion API, where a significant amount of pictures can not be classified, since no face rectangle can be found by the algorithm.

A reason for the bad performance of the fine-tuning approach might be, that the structure is inappropriate for the given data quality. A look at the model's prediction precision (Fig. 11), reveals that some classes seem to be almost randomly assigned, which large numbers of misclassified observations.Additionally to potential problems with the dataset, as described in the previous section, this leads to the conclusion that the fine-tuned model is most probably inaccurately specified for the given problem.

The pictures within the original dataset are already rather small with a size of 48×48 pixels. Stepwise dimension reduction over 16 layers in the fine tuning approach requires the filters in each step to be sufficiently small, to run through the complete network. In this case the filter size in the convolutional layers is throughout the network mainly of size 1×1 (see Tab. 2), which inhibits the useful and desired feature of translation invariance of convolutional layers, but may not be able to detect larger, meaningful features.

An improvement of the estimation results could be achieved in two ways: either by increasing the amount of training data or by improving the model architecture. For modern DL methods, a dataset size of about 36,000 images as in the FER2013 example, can be considered relatively small. One way of increasing the amount of training data is using pattern permutation methods of the given labeled data. Pattern permutation methods include mirroring, turning, distortions or random cropping (Simard et al. 2003). Advantage of this method is, that the whole process can be run completely automatically by machines and the labeling for each of the permutations is already known. Some DL libraries for Python, such as Keras, offer an implemented procedure for permutation. A second approach for extending data is to use an online image search with the respective search terms, which needs more human attention in the selection process, but could result in a better picture quality and increased variability.

On the side of potential model improvements, using an ensemble of models is a often used in practice. A model ensemble estimates several different models, varying in size and design, and combining their results. Idea is, that different models may capture different aspects of the data. For the task of facial expression recognition such model combinations allow state of art prediction accuracy of slightly above 70% (Pramerdorfer and Kampel, 2016).

4 Face value

The following section uses deep learning technique from the previous chapter to interpret facial expressions from live-streams of presentations hosted by companies or official institutions and analyzes their respective effect on the stock market. The underlying hypothesis is that nonverbal communication might be an indicator for the value of information presented, that is more honest and more general than a textual analysis. The potential value, that can be derived from facial expressions, will be denoted as the 'face value', in contrast to the value of speech or words in a textual analysis. Due to the very new nature of this topic, the analysis will have a rather exploratory character. As for today, to best found knowledge, no research has been done on how to link biometrical, facial expression data to finance. The facial expression classifier from the previous section will serve as face measure, which automatically quantifies changes in a person's expression.

The Microsoft Emotion API shows comparable results and a reliable estimation procedure for the given task. Since the self-built and trained model has the restriction of the small input size, which must be exactly 48×48 pixels, the following analysis will be made using the Emotion API in order to get emotional scores. The Emotion API self-selects a face rectangle within each picture, which reduces the amount of time for data preparation and image manipulation massively.

4.1 Requirements on data

An early idea of how to measure a company's face value, was linking company live events to the real-time stock price movement. For an first investigation two different live streamed videos are used and the emotional evaluation is compared to the movement of the stock price. Tick data is used in order to achieve a high enough frequency, which is important since the changes in facial expressions can happen in fractions of a second.

One of the chosen examples in this stage of the analysis was the *Jahresgespräch* of Volkswagen AG from 5th of May 2017. To achieve a more or less continuous stream of emotional scores throughout the press events, for each second of the videos five screen shots are generated and sent to Microsoft Emotion API (FVCcall_API). In return a *.json*-output is generated, containing the emotion-score for eight basic emotions: anger, disgust, contempt, fear, happiness, neutrality, sadness and surprise. Since the Microsoft API is a black-box procedure, it has to be assumed, that the last stage of the image processing procedure is done by a multinomial-logit model or softmax-layer. The final emotion-scores for each emotion can be interpreted as probabilities. This is backed up by the property, that the scores for all eight emotions sum up to one for each picture.

Throughout the course of the press conference substantial variations in the emotional-scores could be observed. For all images sent to the API, the most probable emotion was predicted as neutral, with probabilities between 50 to 80 % while other emotions range in a less than a 0 to 10 % interval (FVCemo_vs_stock). A graphical analysis did not yield any reliable results, same for the analysis of correlations. While some correlations show a weak tendency that the prices of the VW stock moved with some of the emotional-scores (positive for happiness, negative for fear, disgust), others are unclear in evaluation (positive correlation for happiness, negative for surprise). Repeating this procedure for different examples, one of the Apple keynote presentations, yields different results.

A deeper analysis of the previous results is not feasible, imposing some restrictions on the considered press conferences. The instantaneous real-time influence of press releases is not supported by literature. A deeper research on a potential time-lag for this analysis was neglected, due to the following reason: Investigating a face value generates two requirements to the data, first, it is necessary that the speech exists as video, because otherwise no information about facial information can be obtained. Second, the event needs to have a news value. News value means, that the event or press conference contains information that is only released to the public in that moment and has therefore the power to affect stock price movements. Third, the hypothesis is required that not only the informational value is important, but also that it attracts real-time a large enough attention of investors during the event itself, otherwise no reasoning between the video data and the stock movement can be made. Therefore only events that are live broadcasted on TV or as livestream online can be considered.

In the beginning of this analysis, company live events such as speeches of Tim Cook (CEO of Apple Inc.) during Apple's developer conference were used. Another example is a press conference from Volkswagen presenting their future strategy. While the Apple events attract a lot of public attention this cannot be generally said for any press conference which is live-streamed. Moreover, it is not possible to say if there is a direct link to the stock price movements. As for today, to best knowledge, no explicit research has been done in the direction of company live events and investor reactions. The missing connection to the stock market can be linked to the missing news value. Since this kind of events are mainly for advertisement and part of a company's public relations strategy it does not contain enough value for investors. Information that is relevant to investors is still rather based on textual information.

Another major drawback in the analysis is, that apart from potential shortcomings of the chosen events, it was not possible to build up a large enough sample to test any kind of hypothesis. For example Apple hosts only two to three of such events with a variety of topics per year, other major companies even more irregular, some companies only outside of trading hours so that the measurement of a instantaneous effect is impossible. Building up a data set of different companies and different events, has the potential of introducing unwanted and uncontrollable noise. Since at this point the definition of what might be a face value was only very vague and there is no research in any direction to rely on, the use of private company events was rejected.

4.2 European Central Bank press conferences

The European Central Bank (ECB) offers publicly available all major speeches since 1998 in written form. Since several years they use press conferences on a regular basis as an important tool of communication. One of the most important regular meetings of the ECB is the Governing Council, which is responsible, among others, for setting the key interest rates for the eurozone. The Governing Council meets since 2015 every 6 weeks on a Thursday (until 2014: approximately every four weeks and more variation on the weekday), with the following fixed structure: on 13:45 CET a press release is published with the main results. Following shortly after at 14:30 CET the ECB President, currently Mario Draghi, explains the council's decisions in a live-streamed press conference (PC), followed by an question and answer session with attending journalists.

Since the decision is announced during trading hours in all European countries, it can be assumed that the information affects stock markets and for example interest bonds and futures still on the day of the PC. The hypothesis that press releases by the ECB have significant influence on stock markets is supported e.g. by Schmeling and Wagner (2015), Rosa (2011), Hussain (2011) and even across the borders of Europe by Hayo et al. (2010). Fig. 13 illustrates the effect of ECB council meetings and press conferences for DAX30 returns. Peaks right before the press release and during the PC are visible with larger volatilities for the rest of the trading day.

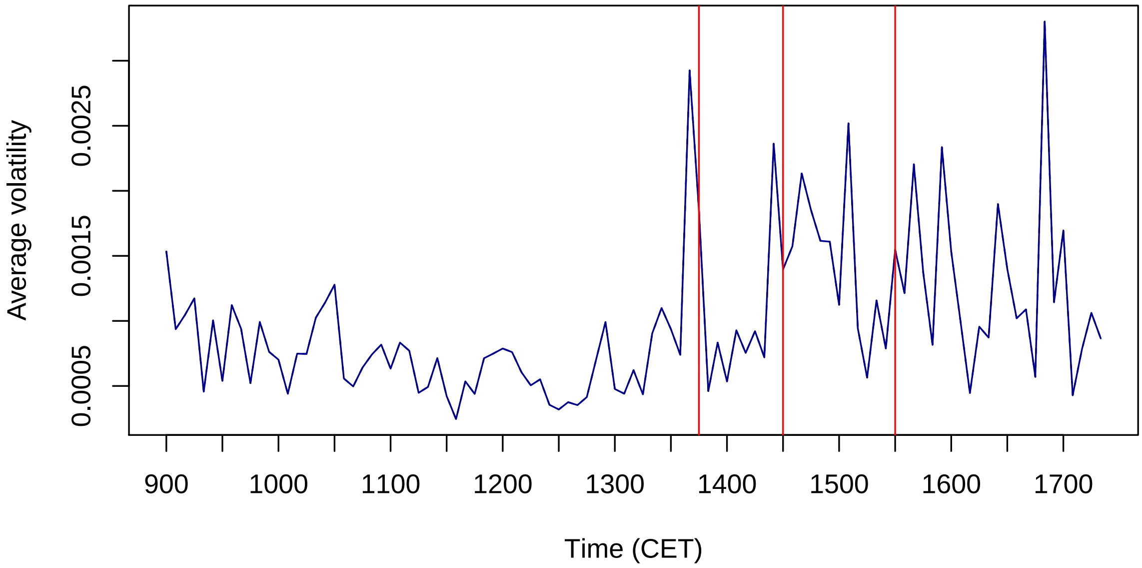


Figure 13: Average volatility of DAX log-returns on days with ECB press announcements (solid) vs. no press announcements (dashed), 2015 - 2017. Vertical lines: Council press release 13:45 CET, begin and approximate end ECB PC 14:30 - 15:30 CET FVCpressdays

All available PCs, which are available as webcasts on youtube.com are used in the following analysis. Webcasts are available since January 2011 until September 2017, which is at the time of this thesis the latest available one. This makes a total of 70 unique PCs. Since older webcasts only cover the part of the president's statement in the beginning of PC only this part is used for every conference. The introductory statement covers about 10-20 minutes with the most important results and explanations. The introductory video, unlike the question and answer session, is shot as a frontal close up of the president's face and is therefore predestined for automated facial expression recognition. For reasons of feasibility two screen shots per second were made and analyzed via Microsoft cognitive service. From a shallow analysis to a higher frequency of screen shots seems not to be highly affecting on the resulting averaged scores.

Overall more than 200 000 pictures were analyzed. Fig. 14 shows the aggregated average emotion per PC between January 2011 and September 2017. Neutrality is left out, since it is the overwhelmingly classified emotion. If all scores were pictured, each bar for each conference would add up to one. Fig. 14 also shows that measuring a person's face with facial expressions is dependent on individual traits: In November 2011 the presidency of the ECB changed from Jean-Claude Trichet to Mario Draghi. Without knowing this fact, the break is visible in the average emotion score.

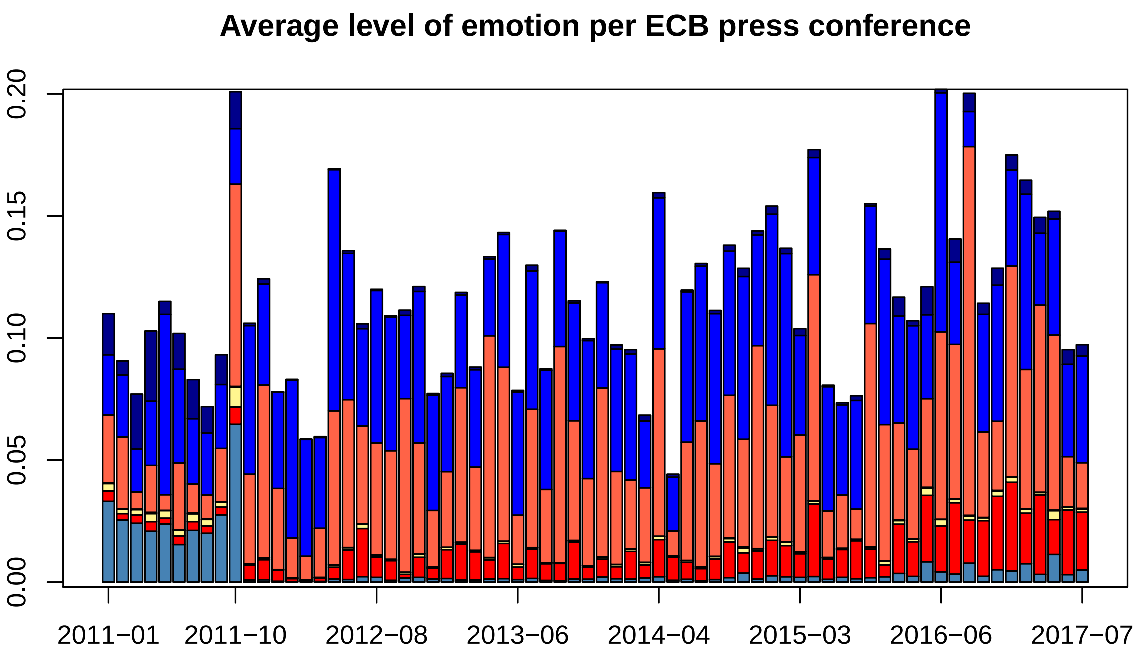


Figure 14: Average aggregated emotional scores for each ECB PC from Jan. 2011 to Sep. 2017
FVCbarplot

4.3 Estimation

In order to determine a value of nonverbal communication during the ECB press conferences, it is compared to other studies which focus on the tonatility of ECB press conferences. Schmeling and Wagner (2015) use the negativity of the speech, derived from a financial dictionary and find that positive (negative) tones during ECB press conferences are associated with an increasing (decreasing) and persistent effect on the stock market until the next press conference takes place.

A face value or nonverbal communication value should have a form, without assuming any specific model, that represents somewhat the relationship

$$r_t = c + \beta\, emotion_t + \varepsilon,$$

with r_t return at a the day of the press conference, $emotion_t$ a kind of signal derived from the facial expression estimation. The parameters c, β, ε being a constant, regression parameter and a general error term. The return will be index data from Euro Stoxx 50. Euro Stoxx 50 consists of the 50 largest companies within the eurozone. It is therefore chosen over Stoxx 50, to have be immediately affected by monetary decisions. Hussain (2011) finds spillover effects of

	(i)	(ii)	(iii)	(iv)	(v)	(vi)	(vii)	(viii)	(ix)
const.	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
lag-return	0.44**	0.46**	0.45**	0.42*	0.38*	0.39*	0.39*	0.39*	0.44**
anger	·	−0.06	−0.03	0.26	0.27	0.22	0.32	0.23	·
contempt	·	·	0.12	0.21	0.31	0.34	0.36	0.38	·
disgust	·	·	·	−2.73	−1.34	−0.85	−1.52	−1.33	·
fear	·	·	·	·	−24.30	−25.51	−24.01	−30.37	·
happiness	·	·	·	·	·	−0.04	−0.04	−0.03	·
sadness	·	·	·	·	·	·	0.05	0.05	·
surprise	·	·	·	·	·	·	·	0.26	·
neutral	·	·	·	·	·	·	·	·	0.00
adj. R^2	0.11	0.10	0.09	0.08	0.09	0.08	0.07	0.05	0.10

Table 3: Results OLS estimation of emotional-scores and return of Eurostoxx50 on PC-days. '*', '**' siginificance at 0.1 % - and 1% - level respectively. FVCEurostoxxOLS

ECB announcements to countries like Switzerland, which are not directly affected by ECB decisions about the the euro, but a stronger effect for members of the monetary union is attestable.

In comparison to sentiment analysis for verbal communication, using facial expressions for nonverbal communication has a specific difficulty: Measuring a sentiment of communication is directly related to the content of speech, meaning that it is very much clear from the words for a human in which direction the sentiment should move. Moreover, there is an intuitive understanding, that if a relationship exists, negative price movements should coincident with a more negative sentiment. Using facial expressions for nonverbal communication leads to the problem, that this relationship is rather indirect, making a detour. It is not clear which emotion might be potentially indirectly affecting prices and in which direction. One of the main considerations is how a signal from facial expressions can be constructed, that capture price movements.

Ordinary least squares

Several model specifications are estimated to detect a potential influence of emotional scores on the end-of-day returns of Euro Stoxx 50 on days where ECB holds press conferences using ordinary least squares estimation (FVCEurostoxxOLS). As a reference model, a model is estimated with one constant and a at level one lagged return parameter, adding the emotional-scores one by one. If the parameters for the emotional-scores equal zero, a linear effect can be neglected.

Tab. 3 shows results for different OLS-regression equations. Only the lagged return parameter

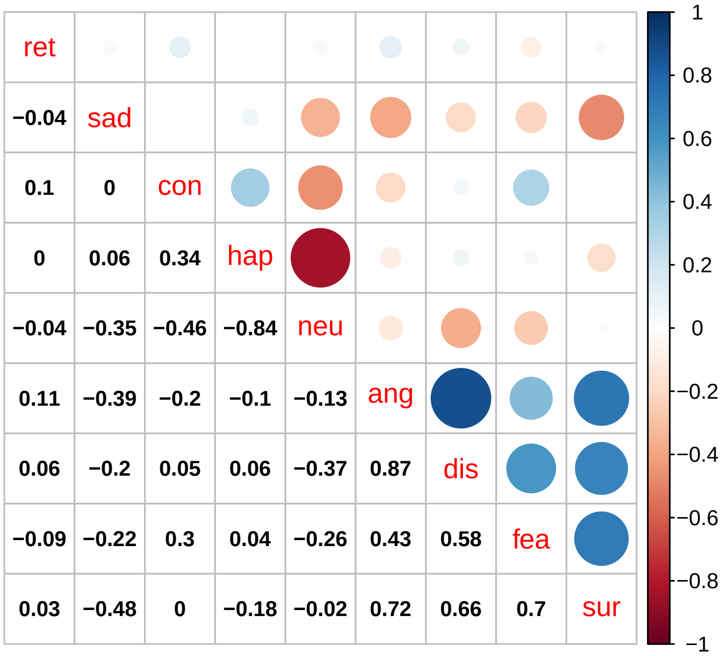


Figure 15: Correlation matrix for emotion-scores and returns of Eursostoxx50 on PC-days FVCcorr

is significant for all models, while the emotional-scores are always insignificant, meaning that no relationship can be assumed. It is not feasible to estimate a model containing all emotional-scores at once, since they sum up to one for each observation, leading to the problem of perfect collinearity. Due to the property, that all scores sum up to one, model (ix) can be interpreted as the overall importance of emotionality throughout the press conference.

Fig. 15 shows correlations between the emotion scores and the respective returns of the day. While there is a significant amount of correlation within the different emotions, no reliable linear connection to the returns can be made.

Partial least squares

Due to the high correlation within the different emotion scores, partial least squares (PLS) estimation is performed. Partial least squares can be used to find essential relations between two matrices (Wold, 1985). Idea is to find m independent components $\mathbf{Z} = \mathbf{z}_1, ..., \mathbf{z}_m$ which fulfill the following relationships simultaneously:

$$\mathbf{X} = \mathbf{Z}\mathbf{P}^\top + \mathbf{E}$$

$$\mathbf{y} = \mathbf{Z}\mathbf{d} + \mathbf{e}.$$

$\mathbf{X}$ is the data matrix of independent variables in this case the emotion scores, $\mathbf{y}$ is the dependent variable, the returns of Euro Stoxx 50 index in this applications. $\mathbf{P}$ is a matrix of loadings and vector $\mathbf{d}$ contains the regression coefficients. $\mathbf{E}$ and $\mathbf{e}$ are the respective error terms.

PLS is related to principal component analysis. In principal component analysis (PCA) the components $\mathbf{Z}$ are chosen to maximize the representational power of the covariance matrix of only $\mathbf{X}$, but without taking their dependence to $\mathbf{y}$ into account. In PLS the covariance matrix between $\mathbf{X}$ and $\mathbf{y}$ is decomposed. That way, the components are chosen to represent the data matrix $\mathbf{X}$, but in order to maximize their representational power towards $\mathbf{y}$.

The number of components in PLS-estimation (FVCEurostoxxPLS) is chosen by cross-validation and set to four. Fig. 16 shows the circle of correlation for the first two components of the PLS estimation. Substantial parts of the variation in the scores of fear, surprise, disgust and anger can be explained by the first two components. For happiness, the variation is only weakly represented by the first two components. But the main question of interest in PLS is, how much of the variability in the dependent variable y can be captured. In this case, y represents the returns of Euro Stoxx 50 on PC-days. Fig. 16 shows that the explanatory power in PLS is rather low. Calculating the model's overall R^2 yields a value of approximately 4 %.

The components of PLS are calculated based on the same data, the model is also evaluated. Therefore a result of 4% can be interpreted as that there is no relationship detectable. A randomization and split into training and testing set, does not yield any other results. In a training and testing split, the resulting R^2 is even lower, also due to the relatively small sample size.

Discriminant analysis

Schmeling and Wagner (2015) show that PC-days with a negative change in sentiment coincident systematically with a negative cumulative return for that day and vice versa for a positive change. A discriminant analysis (DA) is performed on the emotional score data to find a potential positive or negative signal on the returns. DA is a method to separate groups into predefined clusters. Necessary prepocessing step is to separate positive and negative returns on PC-days into two groups with binary encoding (0 - negative return for PC-day, 1 - positive return for PC-day). The sample consists of 70 days of which 27 observations belong to group '0' and 43 observations belong to group '1'.

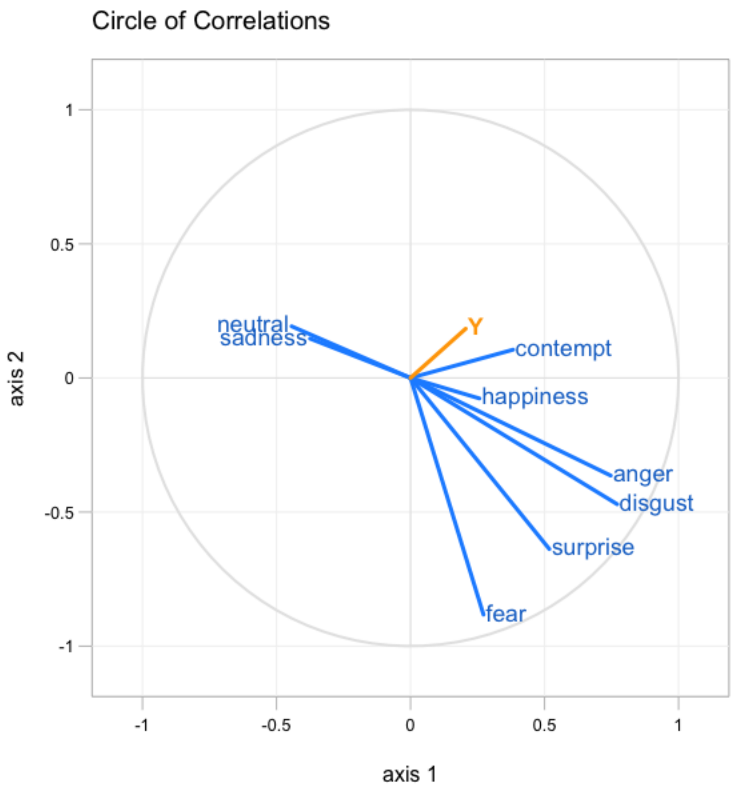


Figure 16: Circle of correlations PLS estimation; Y: dependent variable FVCCorrCircle

A linear DA based on the Bayes rule is performed (Härdle and Simar, 2015, p. 402), because no distributional assumptions can be made (FVCEurostoxxLDA). Analyzing the resulting predictions shows: due to the over-representation of positive returns, the false discovery rate for group one is high (20 over 59), while the recall rate for rising returns is low (7 over 27). Despite having an acceptable accuracy of 46 out of 70 correct classification, the predictive value of the discriminant analysis can be highly doubted. This is also supported by Fig. 17, which shows the histograms and densities arising from the analysis. No substantial differences in the resulting densities are visible.

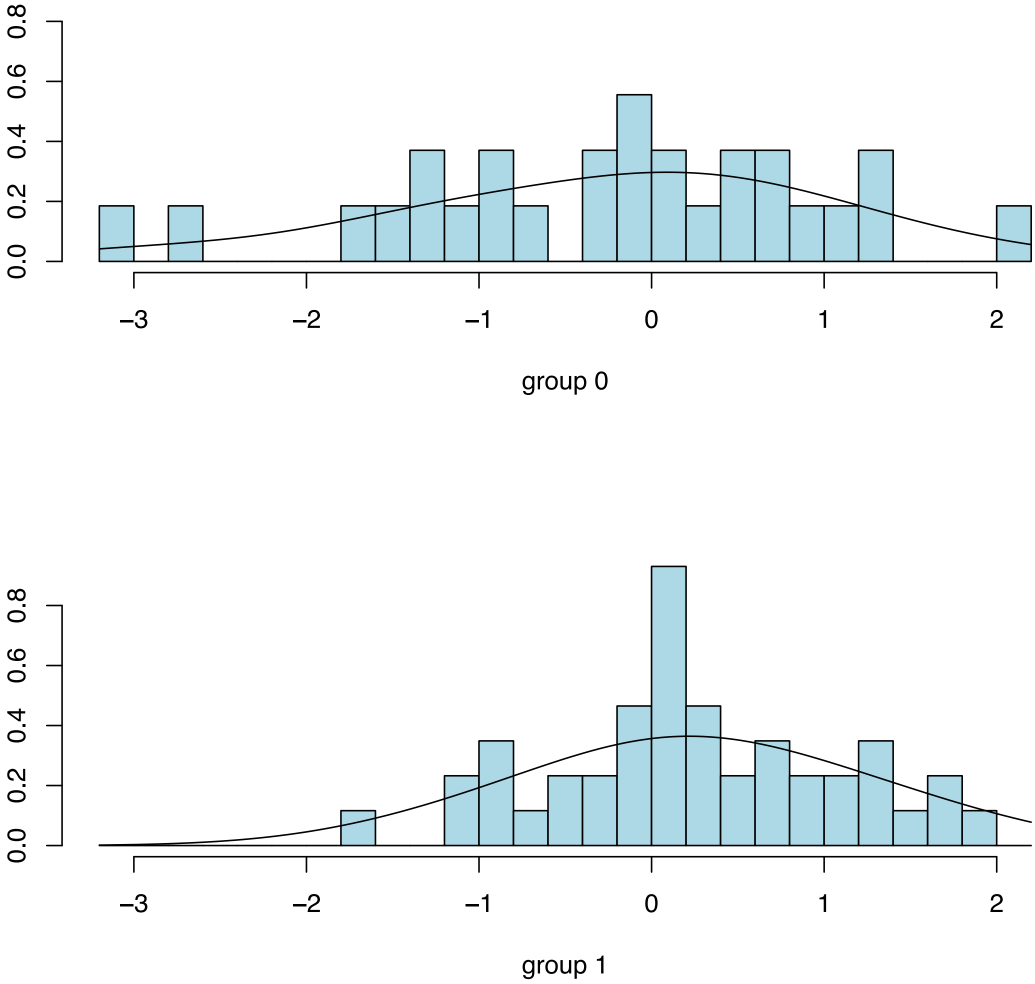


Figure 17: Densities from linear discriminant analysis. Group 0: PC-days with negative returns, Group 1: PC-days with positive returns FVCEurostoxxLDA

5 Conclusion

The broader topic of this thesis is to measure the effect of nonverbal communication on financial data. This effect is denoted as the 'face value' of a company. Nonverbal communication is measured in terms of classifying an image of a face into several basic emotions. Several models are built and compared based on a Kaggle dataset. The best single convolutional neural network has the accuracy of correctly classifying approximately 60% of the images into one over seven classes. This result is comparable to using Microsoft Cognitive Service Emotion API for the same task.

The measurement of the face value has two major problems. First, valuable data is scarce. Second, since there is up to today no research going into this direction, it is unclear on how such a signal on stock prices, generated from emotion-scores, could look like. The first problem is overcome by using data from the European Central Bank, whose regular press conferences provide a good data base. The second problem is treated using several different methods: ordinary least squares, partial least squares and discriminant analysis. Those methods are chosen based on current research of effects of the speech sentiments during ECB press conferences on stock markets. Comparing the results to proven effects of speech sentiment, leads to the result, that with the used methods and approaches no face value of a company can be detected. Side result of this research is, that the facial expression recognition using the described methods is sensitive to personal facial traits, hindering generalizations over different examples.

Since the main problem is to detect a meaningful signal from a person's face, generated by machine intelligence, further research could be done on how to access nonverbal communication. A lot of potential is in a direct comparison to natural language processing. For example, if it is possible to detect a positive or negative sentiment in speech from subtle movements in a person's face. Using facial expressions, expressed as emotions, is only one way to build a metric based on a human face. Other approaches might be considered, for example measuring the speaker's pupil movement. Once there is more substantial knowledge, the range of applications could be broadened, taking the time-lag between signal and financial response into consideration. Also as shown in the research of natural language processing, the effect is not limited to stock prices, it could be extended to other financial and statistical measures such as considering interest rate bonds, which might be of special interest in the context of ECB, or volatilities.

Bibliography

Al-Rfou, R., Alain, G., Almahairi, A., Angermueller, C., Bahdanau, D., Ballas, N., ..., and Zhang, Y. (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.0.

Ba, J. and Caruana, R. (2014). Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*, volume 4. Springer.

Chollet, F. and Others (2015). Keras. https://github.com/fchollet/keras.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314.

Duda, R., Hart, P., and Stork, D. (2001). *Pattern Classification*, volume 2. Wiley-Interscience.

Ekman, P. and Friesen, W. V. (1971). Constants across cultures in the face and emotion. *Journal of Personality and Social Psychology*, 17(2):124–129.

Feng, M., Xiang, B., Glass, M. R., Wang, L., and Zhou, B. (2015). Applying Deep Learning to Answer Selection: A Study and An Open Task. *arXiv preprint arXiv:1508.01585*.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Goodfellow, I. J., Erhan, D., Luc Carrier, P., Courville, A., Mirza, M., Hamner, B., ..., and Bengio, Y. (2015). Challenges in representation learning: A report on three machine learning contests. *Neural Networks*, 64:59–63.

Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. (2013). Maxout Networks. *arXiv preprint arXiv:1302.4389*.

Härdle, W. K. and Simar, L. (2015). *Applied Multivariate Statistical Analysis*. Springer Berlin Heidelberg, Berlin, Heidelberg, 4 edition.

Hayo, B., Kutan, A. M., and Neuenkirch, M. (2010). The impact of U.S. central bank communication on European and pacific equity markets. *Economics Letters*, 108(2):172–174.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Helbing, D. (2015). Societal, Economic, Ethical and Legal Challenges of the Digital Revolution: From Big Data to Deep Learning, Artificial Intelligence, and Manipulative Technologies. *SSRN Electronic Journal*.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, pages 1–18.

Hochreiter, S., Bengio, Y., Frasconi, P., and Schmidhuber, J. (2009). Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies. In *A Field Guide to Dynamical Recurrent Networks*, pages 237–243. John Wiley & Sons.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257.

Huang, G. B., Ramesh, M., Berg, T., and Learned-Miller, E. (2007). Labeled faces in the wild: A database for studying face recognition in unconstrained environments. *University of Massachusetts Amherst Technical Report*, 1:07–49.

Hubel, D. H. and Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160(1):106–154.

Hussain, S. M. (2011). Simultaneous monetary policy announcements and international stock markets response: An intraday analysis. *Journal of Banking and Finance*, 35(3):752–764.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456.

Jarrett, K., Kavukcuoglu, K., Ranzato, M., and LeCun, Y. (2009). What is the best multi-stage architecture for object recognition? *Proceedings of the IEEE International Conference on Computer Vision*, pages 2146–2153.

Kingma, D. P. and Ba, J. L. (2015). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems*, pages 1097–1105.

Kumar, B. S. and Ravi, V. (2016). A survey of the applications of text mining in financial domain. *Knowledge-Based Systems*, 114:128–147.

Kumari, J., Rajesh, R., and Pooja, K. (2015). Facial Expression Recognition: A Survey. *Procedia Computer Science*, 58:486–491.

Lawrence, S., Giles, C., Ah Chung Tsoi, and Back, A. (1997). Face recognition: a convolutional neural-network approach. *IEEE Transactions on Neural Networks*, 8(1):98–113.

LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., and Jackel, L. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4):541–551.

Leshno, M., Lin, V. Y., Pinkus, A., and Schocken, S. (1993). Multilayer Feed Forward Networks With A Nonpolynomial Activation Function Can Approximate Any Function. *Neural Networks*, 6(6):861–867.

Manglik, P. K., Misra, U., Prashant, and Maringanti, H. B. (2004). Facial expression recognition. *IEEE International Conference on Systems, Man and Cybernetics*, 3:2220–2224.

Moniz, A. and de Jong, F. (2014). Predicting the impact of central bank communications on financial market investors' interest rate expectations. In *European Semantic Web Conference*, pages 144–155. Springer.

Nielsen, A. (2015). *Neural Networks and Deep Learning*. Determination Press USA.

Parkhi, O. M., Vedaldi, A., and Zisserman, A. (2015). Deep Face Recognition. *Procedings of the British Machine Vision Conference 2015*, pages (Vol. 1, No. 3, p. 6).

Porter, S. and ten Brinke, L. (2008). Reading Between the Lies: Identifying concealed and falsified emotions in universal facial expressions. *Psychological Science*, 19(5):508–514.

Pramerdorfer, C. and Kampel, M. (2016). Facial Expression Recognition using Convolutional Neural Networks: State of the Art. *arXiv preprint arXiv:1612.02903*.

Qian, N. (1999). On the Momentum Term in Gradient Descent Learning Algorithms The Momentum Term in Gradient Descent. *Neural Networks: The Official Journal of the International Neural Network Society*, 12(1):145–151.

Rojas, R. (1996). The Backpropagation Algorithm. In *Neural Networks*, chapter 7, pages 149–182. Springer, Berlin, Heidelberg.

Rosa, C. (2011). Words that shake traders. The stock market's reaction to central bank communication in real time. *Journal of Empirical Finance*, 18(5):915–934.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

Schmeling, M. and Wagner, C. (2015). Does Central Bank Tone Move Asset Prices? *SSRN Electronic Journal*.

Schmidhuber, J. (2015). Deep Learning in neural networks: An overview. *Neural Networks*, 61:85–117.

Shergill, G. S., Diegel, O., Sarrafzadeh, A., and Shekar, A. (2008). Computerized Sales Assistants: The application of computer technology to measure consumer interest-a conceptual framework. *Journal of Electronic Commerce Research*, 9(2):176.

Silva, L. M., Marques de Sá, J., and Alexandre, L. A. (2008). Data classification with multilayer perceptrons using a generalized error function. *Neural Networks*, 21(9):1302–1310.

Simard, P., Steinkraus, D., and Platt, J. (2003). Best practices for convolutional neural networks applied to visual document analysis. *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, 3:958–963.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Tetko, I. V., Livingstone, D. J., and Luik, A. I. (1995). Neural-Network Studies. 1. Comparison of Overfitting and Overtraining. *Journal of Chemical Information and Computer Sciences*, 35(5):826–833.

Tian, Y., Kanade, T., and Cohn, J. F. (2011). Facial Expression Recognition. In *Handbook of Face Recognition*, pages 487–519. Springer London, London.

Torrey, L. and Shavlik, J. (2009). Transfer Learning. *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques 1*, 1:242–264.

Urban, G., Geras, K. J., Kahou, S. E., Aslan, O., Wang, S., Caruana, R., Mohamed, A., Philipose, M., and Richardson, M. (2016). Do Deep Convolutional Nets Really Need to be Deep and Convolutional? *arXiv preprint arXiv:1603.05691*.

Wold, H. (1985). Partial Least Squares. In *Encyclopedia of Statistical Sciences*. John Wiley & Sons, Inc.

Wolf, L., Hassner, T., and Maoz, I. (2011). Face recognition in unconstrained videos with matched background similarity. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 529–534. IEEE.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? *Advances in Neural Information Processing Systems 27 (Proceedings of NIPS)*, 27:3320–3328.

Yu, Z. and Zhang, C. (2015). Image based Static Facial Expression Recognition with Multiple Deep Network Learning. *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*, pages 435–442.

Zeiler, M. D. (2012). ADADELTA: An Adaptive Learning Rate Method. *arXiv preprint arXiv:1212.5701*.

Zhan, C., Li, W., Ogunbona, P., and Safaei, F. (2008). A Real-Time Facial Expression Recognition System for Online Games. *International Journal of Computer Games Technology*, 2008:10.

Declaration of Authorship

I hereby certify that the thesis I am submitting is entirely my own original work except where otherwise indicated. I am aware of the University's regulations concerning plagiarism, including those regulations concerning disciplinary actions that may result from plagiarism. Any use of the works of any other author, in any form, is properly acknowledged at their point of use.

Berlin, November 24, 2017

Sophie Burgard